

Bu kitaba sığmayan daha neler var!



Karekodu okut, bu kitapla
ilgili EBA içeriklerine ulaş!



ISBN:978-975-11-6422-3



**BU DERS KİTABI MİLLÎ EĞİTİM BAKANLIĞINCA
ÜCRETSİZ OLARAK VERİLMİŞTİR.
PARA İLE SATILAMAZ.**

*Bandrol Uygulamasına İlişkin Usul ve Esaslar Hakkında Yönetmeliğin Beşinci Maddesinin
İkinci Fıkrası Çerçevesinde Bandrol Taşınması Zorunlu Değildir.*

BİLİŞİM TEKNOLOJİLERİ ALANI

MİKRODENETLEYİCİ

11-12

DERS
KİTABI

MESLEKİ VE TEKNİK ANADOLU LİSESİ
BİLİŞİM TEKNOLOJİLERİ ALANI

DERS KİTABI



T.C. MİLLÎ EĞİTİM BAKANLIĞI

MESLEKİ VE TEKNİK ANADOLU LİSESİ

BİLİŞİM TEKNOLOJİLERİ ALANI

MİKRODENETLEYİCİ

DERS KİTABI

Yazarlar

Dr. Fırat YÜCEL

Sefer KAYMAZ

Yalçın TIRAŞ



DEVLET KİTAPLARI

MİLLÎ EĞİTİM BAKANLIĞI YAYINLARI 0000

YARDIMCI VE KAYNAK KİTAPLAR DİZİSİ 0000

Her hakkı saklıdır ve Millî Eğitim Bakanlığına aittir. Kitabın metin, soru ve şekilleri kısmen de olsa hiçbir surette alınıp yayımlanamaz.

HAZIRLAYANLAR

Dil Uzmanı

Erman Erşan YORGANCILAR

Program Geliştirme Uzmanı

Şahinde Seval EZER

Ölçme ve Değerlendirme Uzmanı

Hatice GÜRDİL EGE

Rehberlik Uzmanı

Fatih DÜĞENCİ

Görsel Tasarım Uzmanı

Sermin FIRAT SOYDAN

ISBN: 978-975-11-6422-3

Milli Eğitim Bakanlığı, Talim Terbiye Kurulunun 21.12.2020 gün ve 18433866 sayılı kararı ile ders kitabı olarak kabul edilmiştir.



İSTİKLÂL MARŞI

Korkma, sönmez bu şafaklarda yüzen al sancak;
Sönmeden yurdumun üstünde tüten en son ocak.
O benim milletimin yıldızıdır, parlayacak;
O benimdir, o benim milletimindir ancak.

Çatma, kurban olayım, çehreni ey nazlı hilâl!
Kahraman ırkıma bir gül! Ne bu şiddet, bu celâl?
Sana olmaz dökülen kanlarımız sonra helâl.
Hakkıdır Hakk'a tapan milletimin istiklâl.

Ben ezelden beridir hür yaşadım, hür yaşarım.
Hangi çılgın bana zincir vuracakmış? Şaşarım!
Kükremiş sel gibiyim, bendimi çiğner, aşarım.
Yırtarım dağları, enginlere sığmam, taşarım.

Garbın âfâkını sarmışsa çelik zırhlı duvar,
Benim iman dolu göğsüm gibi serhaddim var.
Ulusun, korkma! Nasıl böyle bir imanı boğar,
Medeniyet dediğin tek dişi kalmış canavar?

Arkadaş, yurduma alçakları uğratma sakın;
Siper et gövdeni, dursun bu hayâsızca akın.
Doğacaktır sana va'dettiği günler Hakk'ın;
Kim bilir, belki yarın, belki yarından da yakın.

Bastığın yerleri toprak diyerek geçme, tanı:
Düşün altındaki binlerce kefensiz yatanı.
Sen şehit oğlusun, incitme, yazıktır, atanı:
Verme, dünyaları alsan da bu cennet vatanı.

Kim bu cennet vatanın uğruna olmaz ki feda?
Şüheda fışkıracak toprağı sıksan, şüheda!
Cânı, cânânı, bütün varımı alsın da Huda,
Etmesin tek vatanımdan beni dünyada cüda.

Ruhumun senden İlahî, şudur ancak emeli:
Değmesin mabedimin göğsüne nâmahrem eli.
Bu ezanlar -ki şehadetleri dinin temeli-
Ebedî yurdumun üstünde benim inlemeli.

O zaman vecd ile bin secde eder -varsa- taşım,
Her cerîhamdan İlahî, boşanıp kanlı yaşım,
Fışkırır ruh-ı mücerret gibi yerden na'sım;
O zaman yükselerek arşa değer belki başım.

Dalgalan sen de şafaklar gibi ey şanlı hilâl!
Olsun artık dökülen kanlarımın hepsi helâl.
Ebediyyen sana yok, ırkıma yok izmihlâl;
Hakkıdır hür yaşamış bayrağımın hürriyet;
Hakkıdır Hakk'a tapan milletimin istiklâl!

Mehmet Âkif Ersoy

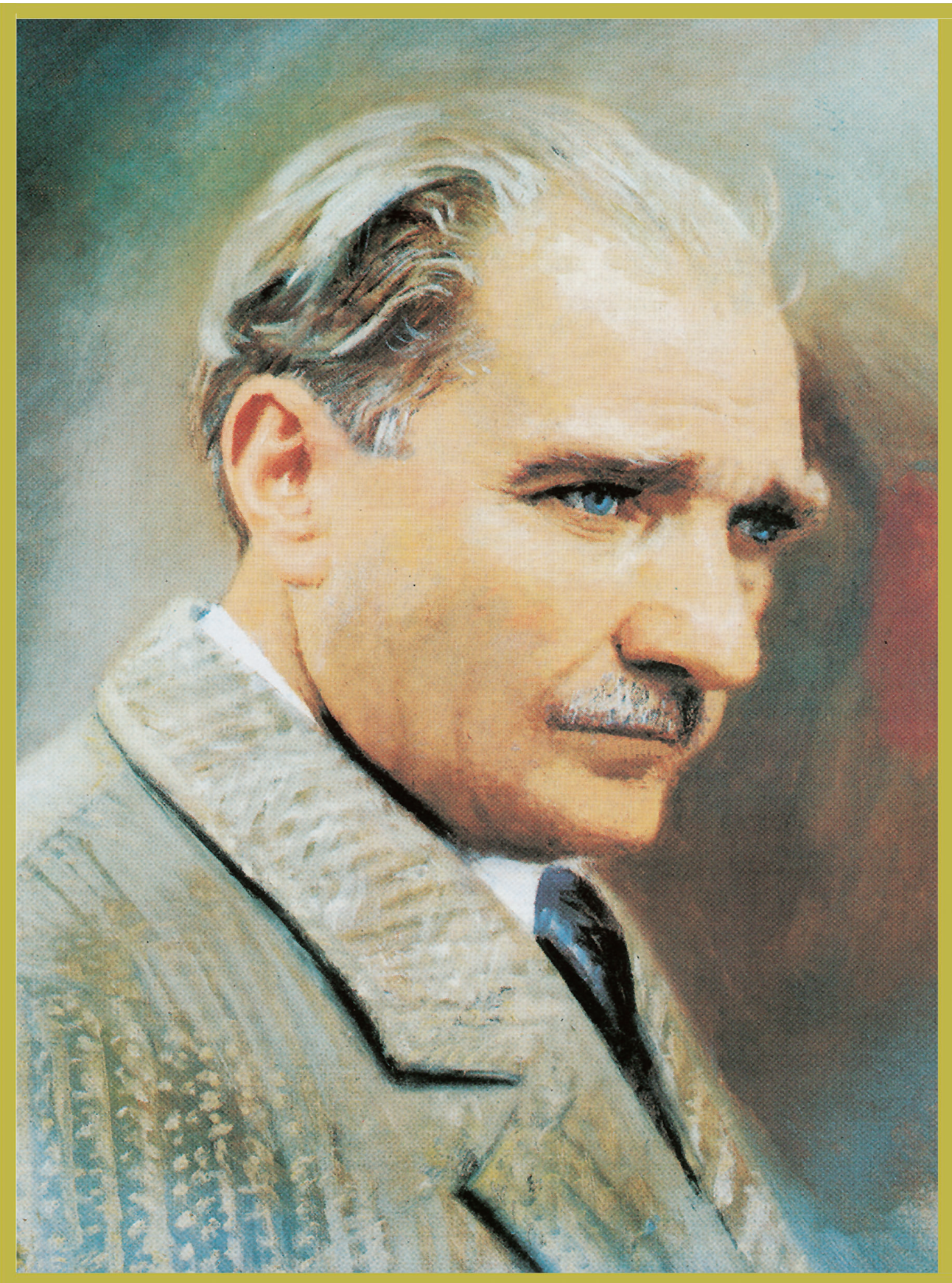
GENÇLİĞE HİTABE

Ey Türk gençliği! Birinci vazifen, Türk istiklâlini, Türk Cumhuriyetini, ilelebet muhafaza ve müdafaa etmektir.

Mevcudiyetinin ve istikbalinin yegâne temeli budur. Bu temel, senin en kıymetli hazinendir. İstikbalde dahi, seni bu hazineden mahrum etmek isteyecek dâhilî ve hâricî bedhahların olacaktır. Bir gün, istiklâl ve cumhuriyeti müdafaa mecburiyetine düşersen, vazifeye atılmak için, içinde bulunacağın vaziyetin imkân ve şeraitini düşünmeyeceksin! Bu imkân ve şerait, çok namüsait bir mahiyette tezahür edebilir. İstiklâl ve cumhuriyetine kastedecek düşmanlar, bütün dünyada emsali görülmemiş bir galibiyetin mümessili olabilirler. Cebren ve hile ile aziz vatanın bütün kaleleri zapt edilmiş, bütün tersanelerine girilmiş, bütün orduları dağıtılmış ve memleketin her köşesi bilfiil işgal edilmiş olabilir. Bütün bu şeraitten daha elîm ve daha vahim olmak üzere, memleketin dâhilinde iktidara sahip olanlar gaflet ve dalâlet ve hattâ hıyanet içinde bulunabilirler. Hattâ bu iktidar sahipleri şahsî menfaatlerini, müstevlîlerin siyasî emelleriyle tevhit edebilirler. Millet, fakr u zaruret içinde harap ve bîtap düşmüş olabilir.

Ey Türk istikbalinin evlâdı! İşte, bu ahval ve şerait içinde dahi vazifen, Türk istiklâl ve cumhuriyetini kurtarmaktır. Muhtaç olduğun kudret, damarlarındaki asil kanda mevcuttur.

Mustafa Kemal Atatürk



MUSTAFA KEMAL ATATÜRK

İÇİNDEKİLER

KİTABIN TANITIMI	13
ÖĞRENME BİRİMİ 1: SAYISAL İŞLEMLER	16
1.1. SAYI SİSTEMLERİ İLE SAYISAL İŞLEMLER	18
1.1.1. Sayı Sistemleri	18
1.1.1.1. Onluk (Decimal) Sayı Sistemleri	19
1.1.1.2. İkilik (Binary) Sayı Sistemleri	19
1.1.1.3. Sekizlik (Octal) Sayı Sistemleri	19
1.1.1.4. Onaltılık (Hexadecimal) Sayı Sistemleri	19
1.1.2. Sayı Sistemlerinde Aritmetik İşlemler	19
1.1.2.1. İkilik (Binary) Sayılarda Toplama İşlemi	20
1.1.2.2. İkilik (Binary) Sayılarda Çıkarma İşlemi	21
1.1.2.3. İkilik (Binary) Sayılarda Çarpma İşlemi	22
1.1.2.4. İkilik (Binary) Sayılarda Bölme İşlemi	23
1.1.2.5. Sekizlik (Octal) Sayılarda Toplama İşlemi	24
1.1.2.6. Sekizlik (Octal) Sayılarda Çıkarma İşlemi	24
1.1.2.7. Onaltılık (Hexadecimal) Sayılarda Toplama İşlemi	25
1.1.2.8. Onaltılık (Hexadecimal) Sayılarda Çıkarma İşlemi	26
1.1.3. Sayı Sistemleri Arasındaki Dönüştürmeler	28
1.1.3.1. Onluk (Decimal) Sayının İkilik (Binary) Sayıya Dönüştürülmesi	28
1.1.3.2. Onluk (Decimal) Sayının Onaltılık (Hexadecimal) Sayıya Dönüştürülmesi	29
1.1.3.3. İkilik (Binary) Sayının Onluk (Decimal) Sayıya Dönüştürülmesi	30
1.1.3.4. Onaltılık (Hexadecimal) Sayının Onluk (Decimal) Sayıya Dönüştürülmesi	31
1.1.3.5. Onaltılık (Hexadecimal) Sayının İkilik (Binary) Sayıya Dönüştürülmesi	32
1.1.3.6. İkilik (Binary) Sayının Onaltılık (Hexadecimal) Sayıya Dönüştürülmesi	33
1.1.3.7. Sekizlik (Octal) Sayının İkilik (Binary) Sayıya Dönüştürülmesi	33
1.1.3.8. İkilik (Binary) Sayının Sekizlik (Octal) Sayıya Dönüştürülmesi	34
1.2. TEMEL LOJİK KAPILARDA MANTIKSAL İŞLEMLER	36
1.2.1. Temel Lojik Kapıların Seçimi	36
1.2.1.1. Temel Lojik Kapılar	37
1.2.1.2. Diğer Lojik Kapılar	42
1.2.2. Temel Lojik Kapılarda İşlemler	47
1.2.2.1. Bolen Çarpma	47
1.2.2.2. Bolen Toplama	48
1.2.2.3. Bolen Kanun ve Kuralları	48
1.2.2.4. Dö Morgın (De Morgan) Teoremi	50
1.2.2.5. Çarpımların Toplamı (Minterm)	51
1.2.2.6. Toplamların Çarpımı (Maxterm)	52
1.2.2.7. Lojik Devreden Lojik İfade Elde Etme	53
1.2.2.8. Lojik İfadeden Lojik Devre Çizimi	55
1.2.2.9. Doğruluk Tablosu Hazırlama	56
1.3. TEMEL LOJİK ENTEGRELERLE DEVRELER	60
1.3.1. Temel Lojik Entegrelerin Seçimi	60
1.3.2. Temel Lojik Devreler Kurma	64
1.3.2.1. Sadeleştirilmiş Lojik İfadeden Devre Kurulması	64
1.3.2.2. Karno (Karnough) Haritası Kullanılarak Devre Kurma	70
ÖLÇME VE DEĞERLENDİRME-1	83
ÖĞRENME BİRİMİ 2: MİKRODENETLEYİCİ PROGRAMLAMA	84

2. MİKRODENETLEYİCİ PROGRAMLAMA	86
2.1. MİKRODENETLEYİCİ VE PROGRAMLAMA	86
2.1.1. Mikro İşlemci	86
2.1.1.1. Mikro İşlemci Temel Birimleri	87
2.1.2. Mikrodenetleyici	88
2.1.2.1. Mikrodenetleyicinin Temel Bileşenleri	89
2.1.2.2. Bellek Organizasyonu Açısından Denetleyici Mimarileri	96
2.1.2.3. Mikrodenetleyici Pin Diyagramı ve Kılıf Yapıları	98
2.1.2.4. Besleme Uçları ve Bağlantıları	100
2.1.2.5. Osilatör Devreleri	101
2.1.2.6. Reset (Sıfırlama) Devresi ve Çeşitleri	107
2.1.3. Uygun Mikrodenetleyicinin Seçimi	111
2.1.3.1. Atmega328P Denetleyicisinin Özellikleri	112
2.1.3.2. Mikrodenetleyici Komut Seti	114
2.1.4. Mikrodenetleyici Programlama İçin Gerekenler	117
2.1.5. Programlama Yazılımının Kurulumu	117
2.1.5.1. Proje Oluşturma	122
2.1.5.2. Yazılım Arayüz Tanıtımı	124
2.1.5.3. Kodun Makine Diline Dönüştürülmesi	127
2.2. MİKRODENETLEYİCİYLE GİRİŞ - ÇIKIŞ KONTROLÜ	129
2.2.1. DDRx Kaydedicisi	131
2.2.2. PORTx Kaydedicisi	131
2.2.3. PINx Kaydedicisi	132
2.2.4. Gecikme İşlemleri	136
2.2.5. Bitwise Operatörler	143
2.2.6. Mikrodenetleyiciyle Buton Kontrolü	150
2.3. MİKRODENETLEYİCİYE PROGRAMI YÜKLEYEREK TEST ETME	158
2.3.1. Programlayıcının Mikrodenetleyiciye Bağlantısı	158
2.3.2. Mikrodenetleyiciye Programın Yüklenmesi	160
2.3.3. Mikrodenetleyicinin Osilatör ve Sigorta Ayarlarını Yapmak	164
2.3.3.1. Low Byte Fuse	167
2.3.3.2. High Byte Fuse	170
2.3.3.3. Extended Byte Fuse	171
ÖLÇME VE DEĞERLENDİRME-2	175
ÖĞRENME BİRİMİ 3: MİKRODENETLEYİCİ İLE ÇEVRE BİRİMLERİ BAĞLAMA	178
3.1. MİKRODENETLEYİCİ İLE TUŞ TAKIMINDAN VERİ OKUMA	180
3.2. MİKRODENETLEYİCİ İLE DISPLAY KONTROLÜ	185
3.2.1. 7 Segment Display (7 Parçalı Gösterge)	185
3.2.2. Matris LED Display	197
3.2.3. Karakter LCD	207
3.2.3.1. Kişiyi Özel LCD Dosyası Oluşturma	211
3.3. MİKRODENETLEYİCİ İLE RÖLE KONTROL UYGULAMALARI	221
3.3.1. Röle Çeşitleri	221
3.3.1.1. Çubuk Röle (Reed Röle)	222
3.3.1.2. Termik Röle	222
3.3.1.3. Katı Hal Rölesi (Solid State Röle)	222
3.3.1.4. Kaçak Akım Koruma Rölesi	223
3.3.1.5. Zaman Rölesi	223
3.3.1.6. Faz Koruma Rölesi	223
3.3.2. Röle Seçimi ve Bağlantısı	224
3.4. MİKRODENETLEYİCİ İLE MOTOR KONTROLÜ	231
3.4.1. DC Motor	231
3.4.1.1. Transistörlü H-Köprü Sürücü Devresi	232

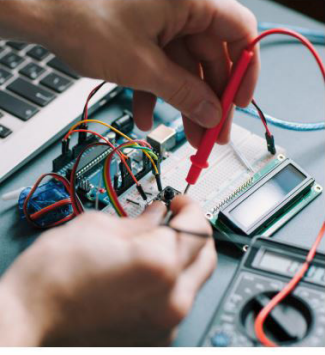
3.4.1.2. DC Motor Sürücü Entegreleri	236
3.4.2. Step Motorlar	241
3.5. MİKRODENETLEYİCİ İLE HABERLEŞME UYGULAMALARI	246
3.5.1. SPI Haberleşmesi	246
3.5.1.1. SPI Çalışma Modları	247
3.5.1.2. SPI Veri Modları	248
3.5.1.3. SPI Kontrol Kaydedicisi	249
3.5.1.4. SPI Veri Aktarım Durumu	250
3.5.2. USART Haberleşmesi	256
3.5.2.1. USART Çalışma Modları	258
3.5.2.2. USART Çerçeve Formatları	259
3.5.2.3. Hata Kontrol Bitlerinin Hesaplanması	260
3.5.2.4. USART Giriş / Çıkış Veri Kaydedicisi n (UDRn)	261
3.5.2.5. USART Kontrol ve Durum İzleme İşlemleri	261
3.5.2.6. USART Baud Rate'in Ayarlanması	264
3.5.2.7. USART Kurulumu	265
3.5.2.8. USART ile Veri Gönderme	265
3.5.2.9. USART ile Veri Alma	266
3.5.3. TWI Haberleşmesi	271
3.5.3.1. BAŞLA ve DUR Durumları	271
3.5.3.2. Adres Paketi Formatı	272
3.5.3.3. Veri Paketi Formatı	273
3.5.3.4. Aktarımda Adres ve Veri Paketlerinin Birleştirilmesi	273
3.5.3.5. TWI Modülünün Yapısı	274
3.5.3.6. TWI Bit Rate Değerinin Belirlenmesi	276
3.5.3.7. TWI İşlemlerinin Kontrolü	276
3.5.3.8. TWI Durumlarının İzlenmesi	277
3.5.3.9. TWI Modülünde Veri Gönderme ve Alma	278
3.5.3.10. TWI Modülünde Adresleme	278
3.5.3.11. TWI Adres Maskeleye	279
3.5.3.12. Master Gönderici Modunda Haberleşme	279
3.5.3.13. Master Alıcı Modunda Haberleşme.....	283
3.5.3.14. Slave Alıcı Modu	286
3.5.3.15. Slave Gönderici Modu	290
ÖLÇME VE DEĞERLENDİRME-3	302
ÖĞRENME BİRİMİ 4: MİKRODENETLEYİCİ İLE ANALOG İŞLEMLER	304
4.1. MİKRODENETLEYİCİ ADC VE DAC ÇEVİRİM KONTROLÜ	306
4.1.1. Analog ve Sayısal İşaretler	306
4.1.2. ADC	311
4.1.3. ADC'nin İç Yapısı	312
4.1.4. Mikrodenetleyici ADC Modülü	313
4.1.4.1. ADC Modülünün Yapısı	313
4.1.4.2. Referans Gerilimi ve ADC Giriş Kanalı Seçimi	315
4.1.4.3. ADC Modülünün Etkinleştirilmesi ve Çevirme İşlemine Başlama	316
4.1.4.4. Kullanılmayan Sayısal Girişlerin Devre Dışı Bırakılması	318
4.1.4.5. ADC Çevirme Sonucunun Bulunması	318
4.1.4.6. ADC Çevirme İşleminin Adımları	319
4.1.4.7. ADC Çevirme İşlemi Tamamlama Kesmesi Kullanma	331
4.1.5. DAC	336
4.1.5.1. Harici DAC Entegresi Kullanma	337
4.1.5.2. Zamanlayıcı/Sayıcı Modülü İle PWM İşareti Üretme	340
4.1.6. Mikrodenetleyici Z/S Modülünün Yapısı	341
4.1.7. Darbe Genişlik Modülasyonu (PWM)	343

4.1.8. Z/S0 Modülü	344
4.1.9. Z/S0 Çalışma Modları	344
4.1.9.1. Normal Mod	344
4.1.9.2. Karşılaştırma Eşleşmesinde Z/S'yi Sıfırlama (CTC) Modu	345
4.1.9.3. Hızlı PWM Modu	346
4.1.9.4. Doğru Faz PWM Modu	347
4.1.10. Ön Ölçekleyici (Prescaler) Kullanımı	348
4.1.11. Z/S0 Modülünde Karşılaştırma Çıkış Modu ve Çalışma Modunun Belirlenmesi	349
4.1.12. Z/S Modülünde Zorlamalı Çıkış Karşılaştırma ve Çalışma Saat Darbesi Seçimi	351
4.1.13. Z/S0 Modülü Z/S Değeri	352
4.1.14. Z/S0 Modülü Çıkış Karşılaştırma Kaydedicileri	352
4.1.15. Z/S0 Modülünde Z/S Kesmelerinin Etkinleştirilmesi	352
4.1.16. Z/S0 Modülünde Z/S Kesme Bayraklarının Kontrolü	353
4.1.17. Z/S0 Modülü Çalıştırma Adımları	353
4.1.18. Z/S1 Modülü	362
4.1.19. Z/S1 Giriş Yakalama Birimi	362
4.1.20. Z/S1 PWM Çalışma Modları	363
4.1.20.1. Hızlı PWM Modu	363
4.1.20.2. Doğru Faz PWM Modu	364
4.1.20.3. Doğru Faz ve Frekans PWM Modu	365
4.1.21. Z/S1 Modülünde Çıkış Karşılaştırma Modu ve Dalga Şekli Üretme Modunun Belirlenmesi	366
4.1.22. Z/S1 Modülünde Giriş Yakalama Ayarları ve Saat Darbesi Kaynağı Seçimi	369
4.1.23. Z/S1 Modülü Zorlamalı Çıkış Karşılaştırma İşlemi	369
4.1.24. Z/S1 Modülü Z/S Kaydedicileri	370
4.1.25. Z/S1 Çıkış Karşılaştırma Kaydedicileri	370
4.1.26. Z/S1 Modülü Giriş Yakalama Kaydedicileri	370
4.1.27. Z/S1 Modülü Kesmeleri Etkinleştirme	371
4.1.28. Z/S1 Modülü Kesme Bayraklarının Kontrolü	371
4.1.29. Z/S1 Modülü Çalıştırma Adımları	372
4.1.30. Z/S2 Modülü	380
4.1.31. Z/S2 Modülünde Karşılaştırma Çıkış Modu ve Çalışma Modunun Belirlenmesi	381
4.1.32. Z/S2 Modülünde Zorlamalı Çıkış Karşılaştırma ve Çalışma Saat Darbesi Seçimi	384
4.1.33. Z/S2 Modülünde Z/S Değeri	384
4.1.34. Z/S2 Modülünde Çıkış Karşılaştırma Kaydedicileri	385
4.1.35. Z/S2 Modülünde Z/S Kesmelerinin Etkinleştirilmesi	385
4.1.36. Z/S2 Modülünde Z/S Kesme Bayraklarının Kontrolü	385
4.1.37. Z/S2 Modülü Çalıştırma Adımları	386
4.2. MİKRODENETLEYİCİ İLE SICAKLIK KONTROLÜ	389
4.2.1. Mikrodenetleyici Dâhili Sıcaklık Algılayıcısı	390
4.2.2. Haricî Sıcaklık Algılayıcısı	393
4.2.2.1. Haricî Sıcaklık Algılayıcı Türleri	393
4.2.2.2. Yarıiletken Sıcaklık Algılayıcıların Kullanımı	395
ÖLÇME VE DEĞERLENDİRME-4	400
KAYNAKÇA	403
YEREL AĞ KAYNAKÇASI VE GÖRSEL KAYNAKÇA	403
ÖĞRENME BİRİMLERİ ÖLÇME VE DEĞERLENDİRME CEVAP ANAHTARLARI	404

KİTABIN TANITIMI

Öğrenme biriminin numarasını gösterir.

3. ÖĞRENME BİRİMİ



MİKRODENETLEYİCİ İLE ÇEVRE BİRİMLERİ BAĞLAMA



Karekod okuyucu ile taratarak resim, video, soru ve çözümleri vb. ilave kaynaklara ulaşabileceğimiz karekod bölümüdür. Detaylı bilgi için <http://kitap.eba.govtr/karekod>

Öğrenme biriminin adını gösterir.

Öğrenme biriminde yer alan konuları gösterir.

KONULAR

- 3.1. MİKRODENETLEYİCİ İLE TUŞ TAKIMINDAN VERİ OKUMA
- 3.2. MİKRODENETLEYİCİ İLE DISPLAY KONTROLÜ
- 3.3. MİKRO DENETLEYİCİ İLE RÖLE KONTROL UYGULAMALARI
- 3.4. MİKRODENETLEYİCİ İLE MOTOR KONTROLÜ
- 3.5. MİKRODENETLEYİCİ İLE HABERLEŞME UYGULAMALARI

NELER ÖĞRENECEKSİNİZ?

- Mikrodenetleyici ile tuş takımından veri okur.
- Mikrodenetleyici ile display kontrolü yapar.
- Mikrodenetleyici ile röle kontrol uygulamaları yapar.
- Mikrodenetleyici ile motor kontrol uygulamaları yapar.
- Mikrodenetleyici ile haberleşme uygulamaları yapar.

ANAHTAR KELİMELER

DC motor, display, haberleşme, H-Köprüsü, Karakter LCD, keypad, master, Matris LED, Mikro işlemci, mikrodenetleyici, motor, röle, slave, SPI, step motor, tuş takımı, TWI, USART, 7 Segment Display.

HAZIRLIK ÇALIŞMALARI

1. Basit bir tuş takımında, hangi tuşa basıldığı sistem tarafından nasıl anlaşılır?
2. Miknahtlanma yöntemi kullanılarak elektrik anahtarları nasıl tasarlanabilir?
3. Mikrodenetleyiciler, diğer çevre birimleri ile (örneğin WIFI modülü, EEPROM vb.) hangi yöntemlerle haberleşir?

Derse başlamadan yapılacak olan hazırlıkları gösterir.

Öğrenme birimindeki önemli kavramları gösterir.

Öğrenme biriminde neler öğreneceğinizi gösteren ön bilgileri gösterir.



ÖLÇME VE DEĞERLENDİRME 4

A)Aşağıda verilen cümlelerin başındaki boşluğa, cümlede yer alan ifade doğru ise "D", yanlış ise "Y" yazınız.

- () Analog işaretler, yalnızca 0 ve 1 değerlerini alır.
- () Örnekleme frekansı (f_s), örnekleme periyodunun (T_s) çarpımına göre tersi alınarak bulunur.
- () Kuantalama işlemi sonucunda, ölçülen işaret belirli seviyelerde değerlendirilir.
- () 10 bitlik bir ADC'nin üretebileceği en yüksek sayısal çıkış kodu değeri 1023'tür.
- () Kesme rutini icra edildikten sonra, program en baştan başlar.
- () Z/S çalışırken kesme kullanılırsa başka komutlar icra edilebilir.

B) Aşağıdaki cümlelerde bulunan boşlukları uygun kelimelerle doldurunuz.

- Bir algılayıcıdan 1 saniyede 50 örnek alınıyorsa örnekleme frekansı Hz'dir.
- Bir işaret, her 50 ms'de bir tanımlı değere sahipse bu işarete denir.
- İlk ADC çevrim işlemi, kurulum işlemleri nedeniyle saat darbesinde yapılır.
- ADC çevirme işlemini başlatmak için adlı kaydedici kullanılır.
- Bir olayın gerçekleştiğini *while(1)* // sürekli döngüsü olmadan otomatik olarak takip etmek için tanımlanabilir.
- Mikrodenetleyici port pinlerinden çekilebilecek en yüksek kaynak (source) akımı, yaklaşık mA'dir.
- Z/S1 modülünde Z/S değeri adlı kaydedicide tutulur.
- TCNTn ve değerlerinin birbirine eşit olması durumunda karşılaştırma eşleşmesi oluşur.

Kazanılan bilgi ve becerilerin her öğrenme birimi sonunda ölçüldüğü çalışmaları gösterir.

KAYNAKÇA

- AEQ Web (2021), Atmega328 16x2 4-Bit LCD Display, Ww.Aeq-Web.Com.
- Altınbaşak O. (2017), Mikrodenetleyiciler ve PIC Programlama
- A. S. Sedra, K. C. Smith (2011), Microelectronic Circuits, 6th Edition, Oxford University Press, Pp. 4-13.
- B. Cincorap (2016), Atmel Programlama 7- Usbasp Kurulumu. <https://bariscincorap.blogspot.com/2016/01/Atmel-Avr-7-Usbasp-Kurulumu.html>
- Bereket Metin, Tekin Engin (2015), Dijital Elektronik
- Bilşim Teknolojileri Alanı Çerçeve Öğretim Programı (2017). Ankara: Millî Eğitim Bakanlığı Yayınları.
- Bilşim Teknolojileri Alanı Ders Bilgi Formu (2020). Ankara: Millî Eğitim Bakanlığı Yayınları.
- D. Hüseyin (2012), Dijital Elektronik
- D. Zhu, T. Siffleet, T. Nunnally, Y. Huang (2021), Analog To Digital Converters, Gatech University, https://ume.gatech.edu/mechatronics_course/ADC_F08.Pdf.
- G. Dökmetaş (2018). C ile AVR Programlama. <http://www.lojikprob.com/avr/c-ile-avr-programlama-60-butun-derslerin-listesi/>
- Microchip Technology (2020), Atmega48a/PA/88A/PA/168A/PA/328/P Megaavr Data Sheet (DS40002061B), Pp. 1-653.
- National Semiconductor (1999), DAC0800/DAC0802 8-Bit Digital-To-Analog Converters Data Sheet (DS005686), Pp. 1-11.
- R. Çevik (2021), Sıcaklık Sensörleri Nedir? Çeşitleri Nelerdir?, <https://www.elektrikport.com/makale-detay/sicaklik-sensorleri-nedir-cesitleri-nelerdir/22055>.
- S. Çiçek (2012). CCS C ile PIC Programlama
- Texas Instruments (2017), LM35 Precision Centigrade Temperature Sensors Data Sheet (SNIS159H), Pp. 1-38.
- M. Yağmurlu, F. Akar(2014). Dijital Elektronik

GENEL AĞ KAYNAKÇASI VE GÖRSEL KAYNAKÇA

<http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=2417>



Karekod, görsel kaynakçasını gösterir.

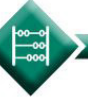


UYGULAMA

Adı:	Sayı Sistemleri ile İlgili Aritmetik İşlemler	No.: 1.1
Amaç:	Sayı sistemleri ile ilgili aritmetik işlemler yapmak.	Süre: 20 dk.

Öğrencilerin edindiği bilgiyi kullanmasını sağlayacak çalışmaları gösterir.

Konuyla ilgili bilgilerin yer aldığı bölümdür.



BİLGİ

CISC (Complex Instruction Set Computer), karmaşık komut kümeli bilgisayar olarak adlandırılır. Bu mimaride yazılan programların verileri işleyebilmesi için çok



SIRA ŞİZDE

Görev: 8 portuna bağlı 7 segment göstergede, 0'dan 9'a kadar olan rakamları birer saniye aralıklar ile önce 0-9 yönünde (ileri) daha sonra 9-0 (geri) yönünde

Konuyu pekiştirmek için öğrencilerin yapması gereken etkinlikleri gösterir.

Konuyla ilişkin ek bilgi ve ipuçlarını gösterir.



NOT

Karşılaştırma eşleşmesi, TCNT2 ile OCRA2'nin karşılaştırılması sonucunda eşit olması durumudur.



ÖRNEK 11

Aşağıda verilen onluk sayıyı ikilik sayıya dönüştürünüz.

$$(51)_{10} = (?)_2$$

Konuyla ilgili örneklerin verildiği bölümdür.

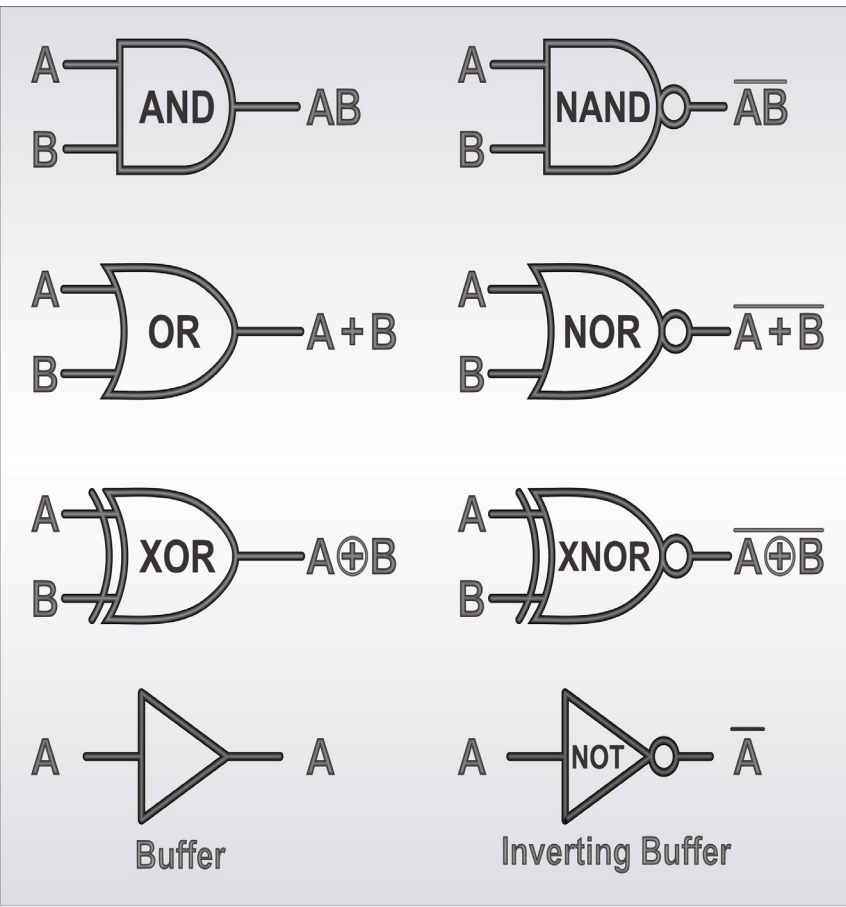
Konunun önemli bölümlerini gösterir.



ÖNEMLİ

Görsel 2.52'deki pencerede /*...*/ ve // sembolleri arasında bulunan metinler, **açıklama satırları** olarak adlandırılır. Bu alanlara yazılan yazılar derleyici tarafından

1. ÖĞRENME BİRİMİ



SAYISAL İŞLEMLER



KONULAR

- 1.1. SAYI SİSTEMLERİ İLE SAYISAL İŞLEMLER
- 1.2. TEMEL LOJİK KAPILARLA MANTIKSAL İŞLEMLER
- 1.3. TEMEL LOJİK ENTEGRELERLE DEVRELER

NELER ÖĞRENECEKSİNİZ?

- Bilgisayar sistemlerinde kullanılan sayı sistemleri
- Sayı sistemlerinin birbirlerine dönüşümleri
- Temel lojik kapılarla işlemler
- Lojik devrelerin seçimi
- Lojik devrelerin kurulumu

ANAHTAR KELİMELER

Sayı sistemleri, temel lojik kapılar, lojik devreler, sadeleştirme, karno haritaları, lojik entegreler.

HAZIRLIK ÇALIŞMALARI

1. Sayı sistemleri denilince aklınıza neler gelmektedir?
2. Mantıksal (lojik) düşünceler, elektronik devre ile nasıl tasarlanabilir?
3. Mantıksal devreler denilince aklınıza neler gelmektedir?

1.1. SAYI SİSTEMLERİ İLE SAYISAL İŞLEMLER

Rakamların tek başına veya çokluğu belirtmek için bir araya getirilmesi ile elde edilen ifadeler **sayı** denir. Sayılar rakamlardan oluşur. Sayma işlemlerinde genellikle onluk sayı sistemi kullanılırken bilgisayar sistemlerinde ise farklı sayı sistemleri de kullanılmaktadır.

1.1.1. Sayı Sistemleri

Sayı sistemleri, sayıları temsil eden simgeler için geliştirilmiş matematiksel bir gösterim sistemidir. Dünya genelinde matematiksel işlemlerde 0-9 arası rakamlardan oluşan **onluk** [decimal (desimal)] sayı sistemi kullanılır. Bilgisayar sistemlerinde ise **ikilik** [binary (binari)], **sekizlik** [octal (oktal)] ve **onaltılık** [hexadecimal (heksadesimal)] sayı sistemleri kullanılır. Görsel 1.1’de sayı sistemleri gösterilmektedir.

Onluk sayı sistemi	İkilik sayı sistemi	Onaltılık sayı sistemi	Sekizlik sayı sistemi
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Görsel 1.1: Sayı sistemleri

1.1.1.1. Onluk (Desimal) Sayı Sistemi

Onluk sayı sistemi; 0, 1, 2, 3, 4, 5, 6, 7, 8 ve 9 rakamlarından oluşur. Sayı sisteminin tabanı ondur. Onluk sistemdeki bir sayının değeri şu formül ile hesaplanır:

$$(ABCD)_{10} = A \times 10^3 + B \times 10^2 + C \times 10^1 + D \times 10^0$$

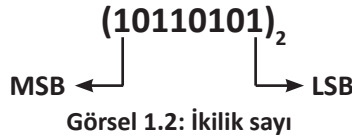
Örneğin, 10 tabanında 341 sayısı,

$3 \times 10^2 + 4 \times 10^1 + 1 \times 10^0 = 300 + 40 + 1 = 341$ şeklinde bulunur.

1.1.1.2. İkilik (Binary) Sayı Sistemi

İkilik sayı sistemi; 0 ve 1 rakamlarından oluşur. Sayı sisteminin tabanı ikidir. İkilik sistemdeki 0 rakamı düşük seviyeyi (LOW, 0 volt veya negatif voltaj) temsil ederken 1 rakamı yüksek seviyeyi (HIGH, 3,3 volt, 5 volt, 12 volt, 24 volt vb.) temsil eder. Her rakam **bit** olarak ifade edilir. Dört bitten oluşan sayılara **nibıl (nibble)**, sekiz bitten oluşan sayılara **bayt (byte)**, iki bayttan oluşan sayılara **vört (word)**, iki vörtten oluşan sayılara **duble vört (double word)** adı verilir.

İkilik sayının en sağında yer alan bite **düşük bit (LSB)**, en solunda yer alan bite **yüksek bit (MSB)** adı verilir. Görsel 1.2’de bir baytlık ikili sayı görülmektedir.



1.1.1.3. Sekizlik (Octal) Sayı Sistemi

Sekizlik sayı sistemi; 0, 1, 2, 3, 4, 5, 6, 7 rakamlarından oluşur. Sayı sisteminin tabanı sekizdir. Görsel 1.3’te sekizlik sayı görülmektedir.

$$(731)_8$$

Görsel 1.3: Sekizlik sayı

1.1.1.4. Onaltılık (Hexadecimal) Sayı Sistemi

Onaltılık sayı sistemi; 0’dan 15’e kadar olan sayılardan oluşmaktadır. Sayı sisteminin tabanı on altıdır. Sıfırdan dokuza kadar olan sayılar rakam ile, ondan on beşe kadar olan sayılar A, B, C, D, E, F harfleri ile gösterilir. Görsel 1.4’te onaltılık sayı görülmektedir.

$$(9AC0)_{16}$$

Görsel 1.4: Onaltılık sayı

1.1.2. Sayı Sistemlerinde Aritmetik İşlemler

Sayı sistemlerindeki aritmetik işlemler toplama, çıkarma, çarpma ve bölme işlemlerinden oluşmaktadır.

1.1.2.1. İkilik (Binary) Sayılarda Toplama İşlemi

İkilik sayı sisteminde toplama işlemi yapılırken Görsel 1.5'te verilen ikilik toplama işlemleri kullanılır. **Toplam** ve **elde** dikkate alınarak toplama işlemi gerçekleştirilir.

Toplanan ikilik sayılar	Toplam	Elde
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1
1 + 1 + 1	1	1

Görsel 1.5: İkilik toplam işlemleri



ÖRNEK 1

Aşağıda verilen ikilik (binary) sayılarda toplama işlemi yapınız.

$$(1110)_2 + (0101)_2 = (?)_2$$

ÇÖZÜM: Toplama işlemi, onluk sayılarda olduğu gibi önce en sağdaki bitten başlanarak (LSB) yapılır.

$$\begin{array}{r}
 \text{11} \\
 \boxed{11}10 \\
 + 0101 \\
 \hline
 10011
 \end{array}$$

- 0 + 1 toplama işlemi sonucu toplama 1 yazılır.
- 1 + 0 toplama işlemi sonucu toplama 1 yazılır.
- 1 + 1 toplama işlemi sonucunda toplama 0 yazılırken elde bir sonraki basamağa aktarılır.
- 1 + 1 + 0 toplama işlemi sonucunda toplama 0 yazılırken elde bir sonraki basamağa aktarılır.
- Toplama işlemi bittiğinden son oluşan elde, toplam sonucuna yazılıp sonuç elde edilir.



SIRA SİZDE

Aşağıda verilen ikilik (binary) sayılarda toplama işlemi yapınız.

$$(1101)_2 + (1110)_2 = (?)_2$$

1.1.2.2. İkilik (Binary) Sayılarda Çıkarma İşlemi

İkilik sayı sisteminde çıkarma işlemi yapılırken Görsel 1.6'da verilen ikilik çıkarma işlemleri kullanılır. Fark ve borç dikkate alınarak çıkarma işlemi gerçekleştirilir. İki sayı çıkarıldığında çıkan sayı eksilen sayıdan büyük ise bir soldaki haneden borç alınır.

Çıkarılan ikilik sayılar	Fark	Borç
0 - 0	0	0
0 - 1	1	1
1 - 0	1	0
1 - 1	0	0

Görsel 1.6: İkilik çıkarma işlemleri



ÖRNEK 2

Aşağıda verilen ikilik (binary) sayılarda çıkarma işlemi yapınız.

$$(101)_2 - (10)_2 = (?)_2$$

ÇÖZÜM: Çıkarma işlemi, onluk sayılarda olduğu gibi önce en sağdaki bitten başlanarak (LSB) yapılır.

$$\begin{array}{r}
 \textcircled{1}01 \\
 - 10 \\
 \hline
 011
 \end{array}$$

2' lik borç alınır

- 1 - 0 çıkarma işlemi sonucu fark 1 olarak yazılır.
- 0 - 1 işleminin yapılabilmesi için soldaki haneden ikilik borç alınır. 2 - 1 çıkarma işlemi sonucunda fark, 1 olarak yazılır.
- En sol hanedeki 1 sayısı borç verdiği için 0 olur ve farka yazılarak işlem tamamlanır.



ÖRNEK 3

Aşağıda verilen ikilik (binary) sayılarda çıkarma işlemi yapınız.

$$(1001)_2 - (0111)_2 = (?)_2$$

$$\begin{array}{r}
 \textcircled{1}001 \\
 - 0111 \\
 \hline
 0010
 \end{array}$$

1 1 2

- 1 - 1 çıkarma işlemi sonucunda fark 0 olarak yazılır.
- 0 - 1 işleminin yapılabilmesi için soldaki haneden ikilik borç alınarak iki bit sağa borç verilir. 2 - 1 işlemi sonucunda fark, 1 olarak yazılır.
- 1 - 1 çıkarma işlemi sonucunda fark, 0 olarak yazılır.
- En sol hanedeki 1 sayısı, borç verdiği için 0 olur ve farka 0 yazılarak işlem tamamlanır.



SIRA SİZDE

Aşağıdaki verilen ikilik (binary) sayılarda çıkarma işlemi yapınız.

$$(10001)_2 - (01111)_2 = (?)_2$$

1.1.2.3. İkilik (Binary) Sayılarda Çarpma İşlemi

İkilik sayı sisteminde çarpma işlemi yapılırken Görsel 1.7’de verilen ikilik çarpım işlemleri kullanılır. Çarpım işlemlerinden sonra **toplam** ve **elde** dikkate alınarak toplama işlemi gerçekleştirilerek işlem tamamlanır.

Çarpılan ikilik sayılar	Sonuç
0 x 0	0
0 x 1	0
1 x 0	0
1 x 1	1

Görsel 1.7: İkilik çarpma işlemleri



ÖRNEK 4

Aşağıda verilen ikilik (binary) sayılarda çarpma işlemi yapınız.

$$(10)_2 \times (10)_2 = (?)_2$$

ÇÖZÜM:

$$\begin{array}{r}
 10 \\
 \times 10 \\
 \hline
 00 \\
 + 10 \\
 \hline
 100
 \end{array}$$



SIRA SİZDE

Aşağıdaki verilen ikilik (binary) sayılarda çarpma işlemi yapınız.

$$\begin{array}{r} 110 \\ \times 11 \\ \hline \end{array}$$

1.1.2.4. İkilik (Binary) Sayılarda Bölme İşlemi

İkilik sayı sisteminde bölme işlemi yapılırken Görsel 1.8’de verilen ikilik bölme işlemleri kullanılır. Bölme işlemi onluk sayılardaki bölme işlemi gibidir.

Bölünen ikilik sayılar	Sonuç
$0 \div 0$	0
$0 \div 1$	0
$1 \div 0$	0
$1 \div 1$	1

Görsel 1.8: İkilik bölme işlemleri



ÖRNEK 5

Aşağıda verilen ikilik (binary) sayılarda bölme işlemi yapınız.

$$(10100)_2 \div (10)_2 = (?)_2$$

ÇÖZÜM:

$$\begin{array}{r} 10100 \mid 10 \\ - 10 \\ \hline 0010 \\ - 10 \\ \hline 000 \end{array}$$

1 sayısı 10 sayısına bölünemediğinden 0'da aşağıya indirilerek bölüme 0 sayısı yazılmıştır.

Son basamaktaki sayı 0 olduğundan bölüme 0 sayısı yazılmıştır.



SIRA SİZDE

Aşağıdaki verilen ikilik (binary) sayılarda bölme işlemi yapınız.

$$\begin{array}{r} 11011 \mid 11 \\ \hline \end{array}$$

1.1.2.5. Sekizlik (Octal) Sayılarda Toplama İşlemi

Sekizlik sayı sisteminde toplama işlemi yapılırken toplanan basamaktaki sayılar sekizi geçiyorsa bir soldaki basamağa elde aktarılır, kalan ise toplanan basamaktaki sayıların toplam hanesine yazılır. Bu işlem en soldaki sayıya kadar sırasıyla uygulanarak toplama işlemi tamamlanır.



ÖRNEK 6

Aşağıda verilen sekizlik (octal) sayılarda toplama işlemi yapınız.

$$(725)_8 + (534)_8 = (?)_8$$

ÇÖZÜM:

$$\begin{array}{r} \text{1} \quad \text{1} \\ 7 \ 2 \ 5 \\ + \ 5 \ 3 \ 4 \\ \hline 1 \ 4 \ 6 \ 1 \end{array}$$

- $5 + 4 = 9$, $9 \div 8$ işleminde **elde 1** ve **kalan 1** olur. Kalan, toplama; elde, bir sol basamağa aktarılır.
- $1 + 3 + 2 = 6$, 6 sayısı sekizden küçük olduğundan elde 0 ve kalan 6 olur.
- $7 + 5 = 12$, $12 \div 8$ işleminde elde 1 ve kalan 4 olur. Kalan, toplama; elde, bir sol basamağa aktarılır. Sol basamakta sayı kalmadığından elde toplam kısmına indirilerek işlem tamamlanır.



SIRA SİZDE

Aşağıda verilen sekizlik (octal) sayılarda toplama işlemi yapınız.

$$(657)_8 + (433)_8 = (?)_8$$

1.1.2.6. Sekizlik (Octal) Sayılarda Çıkarma İşlemi

Sekizlik sayı sisteminde çıkarma işlemi yapılırken eksilen sayı, çıkan sayıdan büyükse sekizlik çıkarma işlemi yapılır. Eksilen sayı, çıkan sayıdan küçükse soldaki haneden bir adet sekizlik borç alınarak çıkarma işlemi yapılır.



ÖRNEK 7

Aşağıda verilen sekizlik (octal) sayılarda çıkarma işlemi yapınız.

$$(704)_8 - (651)_8 = (?)_8$$

ÇÖZÜM:

$$\begin{array}{r} 704 \\ - 651 \\ \hline 033 \end{array}$$

- $4 - 1 = 3$, fark kısmına 3 yazılır.
- Sekizlik sistemde 0 sayısından 5 sayısı çıkmayacağından soldaki basamaktan bir adet sekizlik borç alınır. Yeni durum $8 - 5$ olur, $8 - 5 = 3$ sonucu bulunur ve fark kısmına 3 yazılır.
- Yedi sayısı bir borç verdiği için altına düşer. Yeni durum $6 - 6 = 0$ olur, fark kısmına 0 yazılarak işlem tamamlanır.

**SIRA SİZDE**

Aşağıda verilen sekizlik (octal) sayılarda çıkarma işlemi yapınız.

$$(523)_8 - (345)_8 = (?)_8$$

1.1.2.7. Onaltılık (Hexadecimal) Sayılarda Toplama İşlemi

Onaltılık sayı sisteminde toplama işlemi yapılırken toplanan basamaktaki sayılar on altıyı geçiyorsa bir soldaki basamağa elde aktarılır, kalan ise toplanan basamaktaki sayıların toplam hanesine yazılır. Bu işlem en soldaki sayıya kadar sırasıyla uygulanarak toplama işlemi tamamlanır.

**ÖRNEK 8**

Aşağıda verilen onaltılık (hexadecimal) sayılarda toplama işlemi yapınız.

$$(79A)_{16} + (5C8)_{16} = (?)_{16}$$

ÇÖZÜM:

$$\begin{array}{r} \textcolor{red}{1} \textcolor{red}{1} \\ 79A \\ + 5C8 \\ \hline D62 \end{array}$$

- $A + 8 = 18$, $18 \div 16$ işleminde elde, 1 ve kalan, 2 olur. Kalan, toplama; elde, bir sol basamağa aktarılır.
- $1 + 9 + C = 22$, $22 \div 16$ işleminde elde, 1 ve kalan, 6 olur. Kalan, toplama; elde, bir sol basamağa aktarılır.
- $1 + 7 + 5 = 13$, 13 sayısının onaltılık karşılığı D olduğundan doğrudan toplam kısmına yazılır.



SIRA SİZDE

Aşağıda verilen onaltılık (hexadecimal) sayılarda toplama işlemi yapınız.

$$(EFC)_{16} + (975)_{16} = (?)_{16}$$

1.1.2.8. Onaltılık (Hexadecimal) Sayılarda Çıkarma İşlemi

Onaltılık sayı sisteminde çıkarma işlemi yapılırken eksilen sayı çıkan sayıdan büyükse onaltılık çıkarma işlemi yapılır. Eksilen sayı çıkan sayıdan küçükse soldaki haneden bir adet onaltılık borç alınarak çıkarma işlemi yapılır.



ÖRNEK 9

Aşağıda verilen onaltılık (hexadecimal) sayılarda çıkarma işlemi yapınız.

$$(AD8)_{16} - (7E3)_{16} = (?)_{16}$$

ÇÖZÜM:

$$\begin{array}{r} \textcolor{red}{\overset{16}{\wedge}} \\ A \ D \ 8 \\ - \ 7 \ E \ 3 \\ \hline 2 \ F \ 5 \end{array}$$

- $8 - 3 = 5$, fark kısmına 5 yazılır.
- Onaltılık sistemde D sayısı on üç, E sayısı on dört olduğundan sol basamaktan bir adet onaltılık borç alınır. Yeni durum $16 + D - E$ olur. $16 + 13 - 14 = 15$ sonucu bulunur. On beş sayısının onaltılık sistemdeki karşılığı F olduğundan fark kısmına F yazılır.
- Onaltılık sistemde A sayısı ondur. Borç verdiğiğinden dokuza düşer. Yeni durum $9 - 7 = 2$ olur. Fark kısmına 2 yazılarak işlem tamamlanır.



SIRA SİZDE

Aşağıda verilen onaltılık (hexadecimal) sayılarda çıkarma işlemi yapınız.

$$(B39)_{16} - (34A)_{16} = (?)_{16}$$



UYGULAMA

Adı:	Sayı Sistemleri ile İlgili Aritmetik İşlemler	No.: 1.1
Amaç:	Sayı sistemleri ile ilgili aritmetik işlemler yapmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda sayı sistemleriyle aritmetik işlemleri yapınız.

Kullanılacak Araç Gereç: Tablo 1.1’de belirtilmiştir.

Tablo 1.1: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Kalem	Kurşun	1 adet
Silgi	Kauçuk	1 adet

Uygulama Adımları:

1. Adım: Görsel 1.9’da verilen sayılar ile aritmetik işlemleri görsel üzerinde yapınız.

$\begin{array}{r} 101101 \\ + 111011 \\ \hline \end{array}$	$\begin{array}{r} 110011 \\ - 010110 \\ \hline \end{array}$
$\begin{array}{r} 110 \\ \times 11 \\ \hline \end{array}$	$\begin{array}{r l} 11011 & 11 \\ \hline \end{array}$

Görsel 1.9: Sayı sistemleri ile aritmetik işlemler

2. Adım: Yaptığınız işlemleri öğretmeninize kontrol ettiriniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. İkilik toplama işlemi doğru yapıldı.		
3. İkilik çıkarma işlemi doğru yapıldı.		
4. İkilik çarpma işlemi doğru yapıldı.		
5. İkilik bölme işlemi doğru yapıldı.		

1.1.3. Sayı Sistemleri Arasındaki Dönüştürmeler

Bir sayının başka sayı sistemi içinde işleme alınmak istendiği durumlarda, o sayının işlem yapılacağı sayı sistemine dönüştürülmesi gerekir.

1.1.3.1. Onluk (Decimal) Sayının İkilik (Binary) Sayıya Dönüştürülmesi

Onluk sayı, ikilik sayıya dönüştürürken bölüm, ikiden küçük olana kadar bölme işlemi yapılır. Her bölme işleminden sonra, kalan işaretlenir ve en son bölümden itibaren kalanlar sırayla soldan sağa doğru yazılarak onluk sayının ikilik karşılığı bulunur.

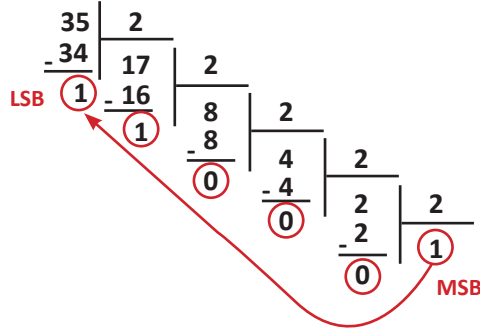


ÖRNEK 10

Aşağıda verilen onluk sayıyı ikilik sayıya dönüştürünüz.

$$(35)_{10} = (?)_2$$

ÇÖZÜM:



$$(35)_{10} = (100011)_2$$

↓MSB
 ↓LSB

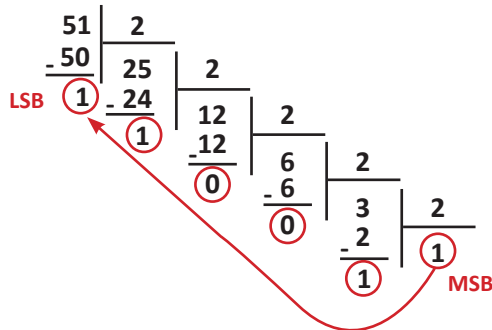


ÖRNEK 11

Aşağıda verilen onluk sayıyı ikilik sayıya dönüştürünüz.

$$(51)_{10} = (?)_2$$

ÇÖZÜM:



$$(51)_{10} = (110011)_2$$

↓MSB
 ↓LSB

**SIRA SİZDE**

Aşağıda verilen işlemi yapınız.

$$(64)_{10} = (?)_2$$

1.1.3.2. Onluk (Decimal) Sayının Onaltılık (Hexadecimal) Sayıya Dönüştürülmesi

Onluk sayı, onaltılık sayıya dönüştürürken bölüm on altıdan küçük oluncaya kadar bölme işlemi yapılır. Her bölme işleminden sonra kalan işaretlenir ve en son bölümden itibaren kalanlar sırayla soldan sağa doğru yazılarak onluk sayının onaltılık karşılığı bulunur.

**ÖRNEK 12**

Aşağıda verilen onluk sayıyı onaltılık sayıya dönüştürünüz.

$$(38)_{10} = (?)_{16}$$

ÇÖZÜM:

$$\begin{array}{r|l} 38 & 16 \\ - 32 & \\ \hline 6 & \end{array}$$

MSB (Most Significant Bit) is 2, LSB (Least Significant Bit) is 6.

$$(38)_{10} = (26)_{16}$$

MSB (Most Significant Bit) is 2, LSB (Least Significant Bit) is 6.

**ÖRNEK 13**

Aşağıda verilen onluk sayıyı onaltılık sayıya dönüştürünüz.

$$(185)_{10} = (?)_{16}$$

ÇÖZÜM:

$$\begin{array}{r|l} 185 & 16 \\ - 16 & \\ \hline 25 & \\ - 16 & \\ \hline 9 & \end{array}$$

MSB (Most Significant Bit) is 11, LSB (Least Significant Bit) is 9.

$$(185)_{10} = (B9)_{16}$$

MSB (Most Significant Bit) is B, LSB (Least Significant Bit) is 9.

Bölümde yer alan 11 sayısının onaltılık karşılığının B olduğu unutulmamalıdır.

**SIRA SİZDE**

Aşağıda verilen işlemi yapınız.

$$(216)_{10} = (?)_{16}$$

1.1.3.3. İkilik (Binary) Sayının Onluk (Decimal) Sayıya Dönüştürülmesi

İkilik sayı, onluk sayıya dönüştürülürken LSB'de bulunan rakamdan itibaren 2^0 dan başlanarak MSB'ye kadar üssü bir artırarak sayı basamak değeri ile çarpılır. Çıkan sonuçlar toplanır ve ikilik sayının onluk karşılığı bulunur.

**ÖRNEK 14**

Aşağıda verilen ikilik sayıyı onluk sayıya dönüştürünüz.

$$(101101)_2 = (?)_{10}$$

ÇÖZÜM: $2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$
 $(1 \ 0 \ 1 \ 1 \ 0 \ 1)_2$

$$1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 + 0 * 2^4 + 1 * 2^5$$

$$1 + 0 + 4 + 8 + 0 + 32 = 45$$

$$(101101)_2 = (45)_{10}$$

Uyarı: $2^0 = 1$ veya herhangi bir sayının sıfıncı üssünün 1 olduğu unutulmamalıdır.

**ÖRNEK 15**

Aşağıda verilen ikilik sayıyı onluk sayıya dönüştürünüz.

$$(11101010)_2 = (?)_{10}$$

ÇÖZÜM: $2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$
 $(1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0)_2$

$$0 * 2^0 + 1 * 2^1 + 0 * 2^2 + 1 * 2^3 + 0 * 2^4 + 1 * 2^5 + 1 * 2^6 + 1 * 2^7$$

$$0 + 2 + 0 + 8 + 0 + 16 + 32 + 64 = 122$$

$$(11101010)_2 = (122)_{10}$$

**SIRA SİZDE**

Aşağıda verilen ikilik sayıyı onluk sayıya dönüştürünüz.

$$(101011)_2 = (?)_{10}$$

1.1.3.4. Onaltılık (Hexadecimal) Sayının Onluk (Decimal) Sayıya Dönüştürülmesi

Onaltılık sayı; onluk sayıya dönüştürülürken en sağda bulunan rakamdan itibaren, 16^0 dan başlanarak en soldaki rakama kadar üssü bir artırarak sayı, basamak değeri ile çarpılır. Çıkan sonuçlar toplanır ve onaltılık sayının onluk karşılığı bulunur.

**ÖRNEK 16**

Aşağıda verilen onaltılık sayıyı onluk sayıya dönüştürünüz.

$$(7B3)_{16} = (?)_{10}$$

ÇÖZÜM: $16^2 \ 16^1 \ 16^0$

$$(7 \ B \ 3)_{16}$$

$$3 * 16^0 + B * 16^1 + 7 * 16^2$$

$$3 + 11 * 16 + 7 * 256$$

$$3 + 176 + 1792 = 1971$$

$$(7B3)_{16} = 1971$$

**ÖRNEK 17**

Aşağıda verilen onaltılık sayıyı onluk sayıya dönüştürünüz.

$$(26EA)_{16} = (?)_{10}$$

ÇÖZÜM: $16^3 \ 16^2 \ 16^1 \ 16^0$

$$(2 \ 6 \ E \ A)_{16}$$

$$A * 16^0 + E * 16^1 + 6 * 16^2 + 2 * 16^3$$

$$10 + 14 * 16 + 6 * 256 + 2 * 4096$$

$$10 + 224 + 1536 + 8192 = 9962$$

$$(26EA)_{16} = 9962$$

**SIRA SİZDE**

Aşağıda verilen onaltılık sayıyı onluk sayıya dönüştürünüz.

$$(15B)_{16} = (?)_{10}$$

1.1.3.5. Onaltılık (Hexadecimal) Sayının İkilik (Binary) Sayıya Dönüştürülmesi

Onaltılık sayı, ikilik sayıya dönüştürülürken her rakamın karşılık geldiği ikili sayı, dört bit olacak şekilde bulunur. İkilik sayı, onaltılık sayının basamağına karşılık gelecek şekilde dizilir. Böylece onaltılık sayının ikilik karşılığı bulunur.

**ÖRNEK 18**

Aşağıda verilen onaltılık sayıyı ikilik sayıya dönüştürünüz.

$$(1A3)_{16} = (?)_2$$

ÇÖZÜM: $(\underline{1} \ \underline{A} \ \underline{3})_{16}$

0001 1010 0011

$$(1A3)_{16} = (\underline{000110100011})_2$$

MSB kısmındaki 3 adet 0 silinerek sayının son hâli bulunur.

$$(1A3)_{16} = (110100011)_2$$

**ÖRNEK 19**

Aşağıda verilen onaltılık sayıyı ikilik sayıya dönüştürünüz.

$$(F4C6)_{16} = (?)_2$$

ÇÖZÜM: $(\underline{F} \ \underline{4} \ \underline{C} \ \underline{6})_{16}$

1111 0100 1100 0110

$$(F4C6)_{16} = (\underline{1111010011000110})_2$$

**SIRA SİZDE**

Aşağıda verilen onaltılık sayıyı ikilik sayıya dönüştürünüz.

$$(A5D7)_{16} = (?)_2$$

1.1.3.6. İkilik (Binary) Sayının Onaltılık (Hexadecimal) Sayıya Dönüştürülmesi

İkilik sayı, onaltılık sayıya dönüştürülürken LSB'den başlamak üzere ikilik sayı, dört bitlik gruplara ayrılır. İkilik sayı dördün katları değilse MSB tarafına, dördün katı olacak şekilde sıfır rakamı eklenir. Her grubun onaltılık karşılığı bulunur. Onaltılık sayılar dört bitlik grupların yerine yazılır. Böylece ikilik sayının onaltılık karşılığı bulunur.



ÖRNEK 20

Aşağıda verilen ikilik sayıyı onaltılık sayıya dönüştürünüz.

$$(1110110001)_2 = (?)_{16}$$

ÇÖZÜM: $(\underline{0011} \underline{1011} \underline{0001})_2$

$$\begin{array}{ccc} \swarrow & \downarrow & \searrow \\ 3 & B & 1 \end{array}$$

$$(1110110001)_2 = (3B1)_{16}$$



ÖRNEK 21

Aşağıda verilen ikilik sayıyı onaltılık sayıya dönüştürünüz.

$$(1110110001000011)_2 = (?)_{16}$$

ÇÖZÜM: $(\underline{1110} \underline{1100} \underline{0100} \underline{0011})_2$

$$\begin{array}{cccc} \swarrow & \swarrow & \swarrow & \swarrow \\ E & C & 4 & 3 \end{array}$$

$$(1110110001000011)_2 = (EC43)_{16}$$



SIRA SİZDE

Aşağıda verilen ikilik sayıyı onaltılık sayıya dönüştürünüz.

$$(111110001011011)_2 = (?)_{16}$$

1.1.3.7. Sekizlik (Octal) Sayının İkilik (Binary) Sayıya Dönüştürülmesi

Sekizlik sayı, ikilik sayıya dönüştürülürken her rakamın karşılık geldiği ikili sayı, üç bit olacak şekilde bulunur. İkilik sayı, sekizlik sayının basamağına karşılık gelecek şekilde dizilir. Böylece sekizlik sayının ikilik karşılığı bulunur.

**ÖRNEK 22**

Aşağıda verilen sekizlik sayıyı ikilik sayıya dönüştürünüz.

$$(273)_8 = (?)_2$$

ÇÖZÜM: $(\underline{2} \ \underline{7} \ \underline{3})_8$

$$\begin{array}{ccc} \swarrow & \downarrow & \searrow \\ 010 & 111 & 011 \end{array}$$

$$(273)_8 = (\underline{010111011})_2$$

MSB kısmındaki 1 adet 0 silinerek sayının son hâli bulunur.

$$(273)_8 = (10111011)_2$$

**SIRA SİZDE**

Aşağıda verilen sekizlik sayıyı ikilik sayıya dönüştürünüz.

$$(654)_8 = (?)_2$$

1.1.3.8. İkilik (Binary) Sayının Sekizlik (Octal) Sayıya Dönüştürülmesi

İkilik sayı, sekizlik sayıya dönüştürülürken LSB'den başlamak üzere ikilik sayı, üç bitlik gruplara ayrılır. İkilik sayıdaki rakamlar sayısı, üçün katları değilse MSB tarafına üçün katı olacak şekilde sıfır rakamı eklenir. Her grubun sekizlik karşılığı bulunur. Sekizlik sayılar üç bitlik grupların yerine yazılır. Böylece ikilik sayının sekizlik karşılığı bulunur.

**ÖRNEK 23**

Aşağıda verilen ikilik sayıyı sekizlik sayıya dönüştürünüz.

$$(1110110001)_2 = (?)_8$$

ÇÖZÜM: $(\underline{001110110001})_2$

$$\begin{array}{cccc} \swarrow & \downarrow & \downarrow & \searrow \\ 1 & 6 & 6 & 1 \end{array}$$

$$(1110110001)_2 = (1661)_8$$

**SIRA SİZDE**

Aşağıda verilen işlemi yapınız.

$$(111101100)_2 = (?)_8$$

**UYGULAMA**

Adı:	Sayı Sistemlerini Birbirlerine Dönüştürme	No.: 1.2
Amaç:	Sayı sistemleri ile dönüştürmeler yapmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda sayı sistemleri arasında dönüştürme işlemleri yapınız.

Kullanılacak Araç Gereç: Tablo 1.2’de belirtilmiştir.

Tablo 1.2: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Kalem	Kurşun	1 adet
Silgi	Kauçuk	1 adet

Uygulama Adımları:

1. Adım: Görsel 1.10’da verilen sayılar ile dönüştürme işlemlerini görsel üzerinde yapınız.

$(101111)_2 = (?)_{10}$	$(78)_{16} = (?)_{10}$
$(123)_{10} = (?)_2$	$(622)_{10} = (?)_{16}$
$(B2A)_{16} = (?)_2$	$(1111010101)_2 = (?)_{16}$

Görsel 1.10: Sayı sistemleri arasında dönüştürme

2. Adım: Yaptığınız işlemleri öğretmeninize kontrol ettiriniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. İkilik sayı sisteminden onluk sayı sistemine dönüştürme işlemi doğru yapıldı.		
3. Onaltılık sayı sisteminden onluk sayı sistemine dönüştürme işlemi doğru yapıldı.		
4. Onluk sayı sisteminden ikilik sayı sistemine dönüştürme işlemi doğru yapıldı.		
5. Onluk sayı sisteminden onaltılık sayı sistemine dönüştürme işlemi doğru yapıldı.		
6. Onaltılık sayı sisteminden ikilik sayı sistemine dönüştürme işlemi doğru yapıldı.		
7. İkilik sayı sisteminden onaltılık sayı sistemine dönüştürme işlemi doğru yapıldı.		

1.2. TEMEL LOJİK KAPILARLA MANTIKSAL İŞLEMLER

Lojik kapılar, girişlerine uygulanan lojik değişkenlerle işlem yapan ve bu işlem sonucunu devrenin lojik çıkışında gösteren elektronik devrelerdir. Lojik kapıların yapısında BJT transistör, MOSFET, diyot, direnç vb. elektronik devre elemanları kullanılır. Yapılarında BJT transistör kullanılan lojik kapılar, **TTL (Transistör-Transistör Lojik)** serisi olarak adlandırılırken yapılarında MOSFET kullanılan lojik kapılar, **CMOS (Tümleşik Metal Oksit Yarıiletken)** serisi lojik kapılar olarak adlandırılır. Lojik kapılar, entegre devreler içerisinde genelde birden fazla sayıda olacak şekilde kılıflanır. TTL serisi lojik kapı entegreleri, 74xx olarak kılıflanırken CMOS serisi lojik kapı entegreleri, 40xx olarak kılıflanır.

Lojik kapılar, sayısal (dijital) elektronik sistemlerinde aritmetik ve mantık işlemlerini yapabilen temel birimlerdir. Basit anlamda programlama elemanları olarak kullanılmaktadır. Programlama, elektriksel bir anahtarlama olarak düşünülürse her lojik kapının anahtarlarla yapılmış eşdeğer devresi çizilebilir. Lojik kapıların anahtar, lamba, direnç, pil vb. ile çizilerek oluşturulan devresine **elektriksel eş değer devresi** denir. Lojik kapılar girişlerine uygulanan lojik 1 [5 volt (HIGH)] ve lojik 0 [0 volt (LOW)] değişkenleri ile işlem yapar.

Lojik kapıların girişleri ile çıkışı arasındaki durum, matematiksel bir ifade ile gösterilir. Bu ifadeye **Bolen (Boolean)** ifadesi adı verilir. Genelde **Q** harfi ile gösterilir. Lojik kapının girişlerine uygulanan lojik değişkenler ile bolen ifadesi doğrultusunda lojik çıkıştan farklı lojik değerler elde edilir. Elde edilen lojik değerlerin yazıldığı tabloya, **doğruluk tablosu** denir.

1.2.1. Temel Lojik Kapıların Seçimi

Lojik devrelerin tasarımında **VE kapısı (AND gate)**, **VEYA kapısı (OR gate)** ve **DEĞİL kapısı (NOT gate)** temel lojik kapılar olarak kabul edilir. Bunun dışında kullanılan kapılar, bu üç kapıdan elde edilmektedir.

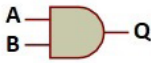
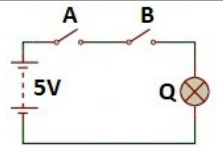
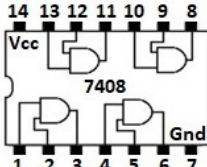
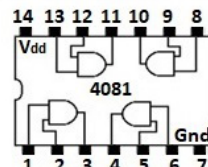
1.2.1.1. Temel Lojik Kapılar

Lojik devrelerde mantıksal çarpma, mantıksal toplama ve mantıksal değil alma işlemlerinde kullanılan temel lojik kapılardır. Bunlar sırasıyla şu başlıklar altında toplanır.

VE Kapısı (AND Gate)

VE kapısı, genellikle iki adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişlerine uygulanan lojik bilgilerin mantıksal çarpımını, lojik çıkışa aktarır. Girişlerin ikisi de lojik 1 ise çıkış lojik 1 olur. Diğer durumlarda çıkış, lojik 0 olur. Bolen ifadesi **$Q = A.B$ (A çarpı B)** olarak ifade edilir.

Görsel 1.11'de VE kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.

Sembolü	Boolean ifadesi	Doğruluk tablosu	Elektriksel eşdeğer devresi																		
	$Q=A.B$	<table><tr><th colspan="2">GİRİŞLER</th><th>ÇIKIŞ</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	GİRİŞLER		ÇIKIŞ	A	B	Q	0	0	0	0	1	0	1	0	0	1	1	1	
GİRİŞLER		ÇIKIŞ																			
A	B	Q																			
0	0	0																			
0	1	0																			
1	0	0																			
1	1	1																			
Entegre yapısı																					
<div><div><p>7408</p><p>TTL</p></div><div><p>4081</p><p>CMOS</p></div></div>																					

Görsel 1.11: VE kapısı



UYGULAMA

Adı:	VE Kapısı ile Mantıksal Çarpma İşlemleri	No.: 1.3
Amaç:	VE kapısı ile mantıksal çarpma işlemleri yapmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda VE kapısı uygulama devresini elektronik çizim programında çizip çalıştırınız.

Kullanılacak Araç Gereç: Tablo 1.3'te belirtilmiştir.

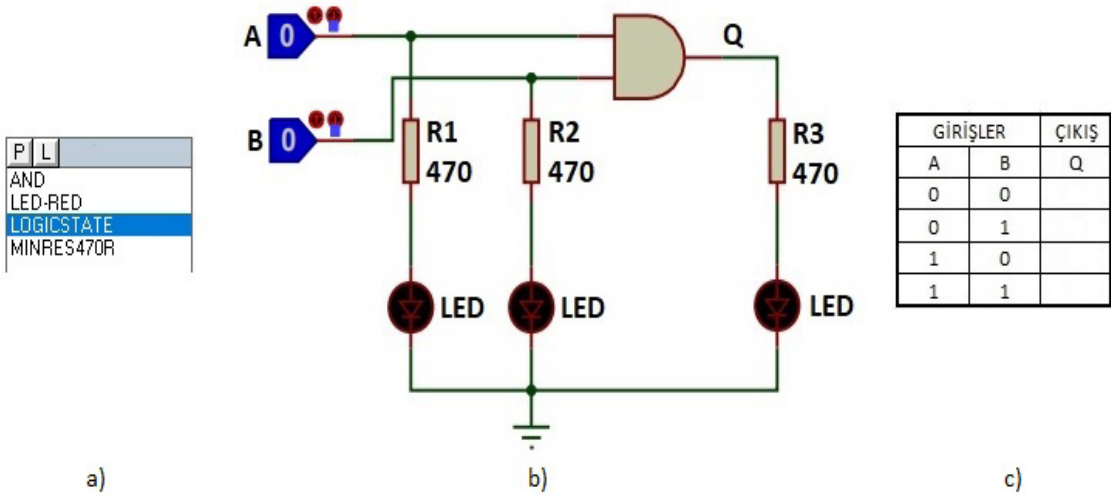
Tablo 1.3: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Elektronik şema çizim programı	Windows tabanlı	1 adet

Uygulama Adımları:

1. Adım: Elektronik devre çizim programını çalıştırınız.

2. Adım: Görsel 1.12:b' de verilen VE kapısı uygulama devresini, Görsel 1.12:a' da gösterilen "DEVICES" penceresindeki devre elemanlarını kullanarak elektronik çizim programında çiziniz.



Görsel 1.12: a) Uygulama devre elemanları Görsel 1.12: b) Devre şeması Görsel 1.12: c) Doğruluk tablosu

3. Adım: A ve B lojik girişlerine, sırasıyla $(00)_2$, $(01)_2$, $(10)_2$ ve $(11)_2$ ikilik bilgilerini uygulayıp, Q çıkışına bağlı LED'in ışık verdiği durumlara 1, ışık vermediği durumlara 0 yazarak Görsel 1.12:c' de verilen doğruluk tablosunu doldurunuz.

4. Adım: Doğruluk tablosuna yazdığınız değerleri öğretmeninize kontrol ettiriniz.

5. Adım: Çizim programı kütüphanesinden 4081 entegresi alarak devreyi tekrar çizip çalıştırınız ve adım 3'ü tekrarlayınız. AND kapısı ile 4081 entegresinin doğruluk tablosunu karşılaştırınız.

6. Adım: Çalışmanız bitince bilgisayarı kapatınız.

7. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız diğer sayfadaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Elektronik çizim programı kütüphanesinden devrede kullanılacak elemanlar doğru seçildi.		
3. Elektronik çizim programında verilen devre doğru çizildi.		
4. Doğruluk tablosu doğru dolduruldu.		
5. Temizliğe dikkat edildi.		

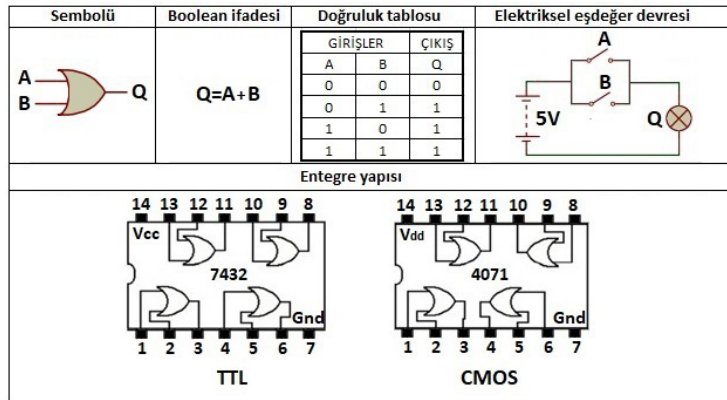


SIRA SİZDE

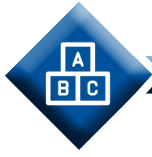
Uygulama 1.3'te verilen lojik devreyi 7408 entegresini kullanarak çizip çalıştırınız. Devrenin doğruluk tablosunu hazırlayınız. VE kapısı ve 4081 entegresi kullanarak hazırladığınız doğruluk tabloları ile 7408 entegresiyle hazırladığınız doğruluk tablosu arasında fark olup olmadığını tespit ediniz.

VEYA Kapısı (OR Gate)

VEYA kapısı, genellikle iki adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişlerine uygulanan lojik bilgilerin mantıksal toplamını lojik çıkışa aktarır. Girişlerin ikisi de lojik 0 ise çıkış lojik 0 olur. Diğer durumlarda çıkış lojik 1 olur. Bolen ifadesi $Q = A + B$ (**A toplam B**) olarak ifade edilir. Görsel 1.13'te VEYA kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.



Görsel 1.13: VEYA kapısı



UYGULAMA

Adı:	VEYA Kapısı ile Mantıksal Toplama İşlemleri	No.: 1.4
Amaç:	VEYA kapısı ile mantıksal toplama işlemleri yapmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda VEYA kapısı uygulama devresini elektronik çizim programında çizip çalıştırınız.

Kullanılacak Araç Gereç: Tablo 1.4’te belirtilmiştir.

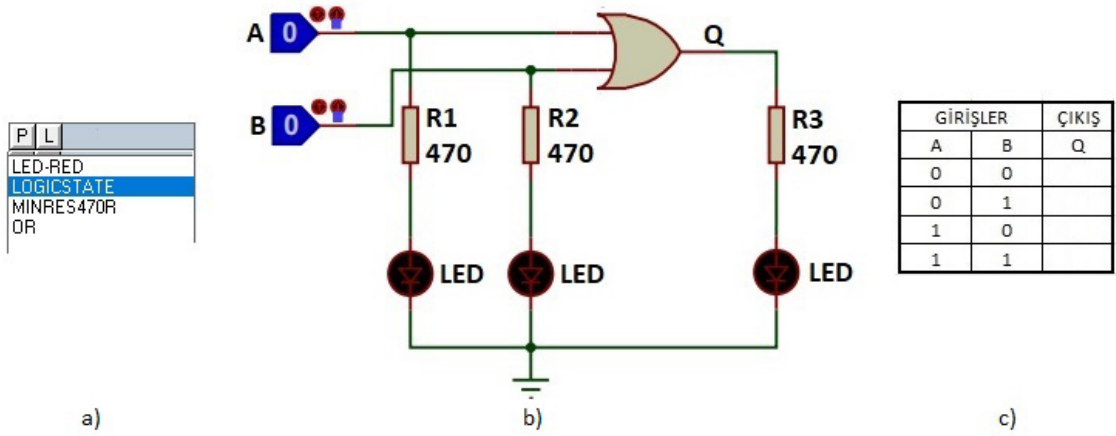
Tablo 1.4: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Elektronik şema çizim programı	Windows tabanlı	1 adet

Uygulama Adımları:

1. Adım: Elektronik devre çizim programını çalıştırınız.

2. Adım: Görsel 1.14:b’ de verilen VEYA kapısı uygulama devresini, Görsel 1.14:a’ da gösterilen “DEVICES” penceresindeki devre elemanlarını kullanarak elektronik çizim programında çiziniz.



Görsel 1.14: a) Uygulama devre elemanları Görsel 1.14: b) Devre şeması Görsel 1.14: c) Doğruluk tablosu

3. Adım: A ve B lojik girişlerine, sırasıyla $(00)_2$, $(01)_2$, $(10)_2$ ve $(11)_2$ ikilik bilgilerini uygulayıp, Q çıkışına bağlı LED’in ışık verdiği durumlara 1, ışık vermediği durumlara 0 yazarak Görsel 1.14:c’ de verilen doğruluk tablosunu doldurunuz.

4. Adım: Doğruluk tablosuna yazdığınız değerleri atölye öğretmeninize kontrol ettiriniz.

5. Adım: Çizim programı kütüphanesinden 4071 entegresi alarak devreyi tekrar çizip çalıştırınız ve adım 3'ü tekrarlayınız. VEYA kapısı ile 4071 entegresinin doğruluk tablosunu karşılaştırınız.

6. Adım: Çalışmanız bitince bilgisayarı kapatınız.

7. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Elektronik çizim programı kütüphanesinden devrede kullanılacak elemanlar doğru seçildi.		
3. Elektronik çizim programında verilen devre doğru çizildi.		
4. Doğruluk tablosu doğru dolduruldu.		
5. Temizliğe dikkat edildi.		



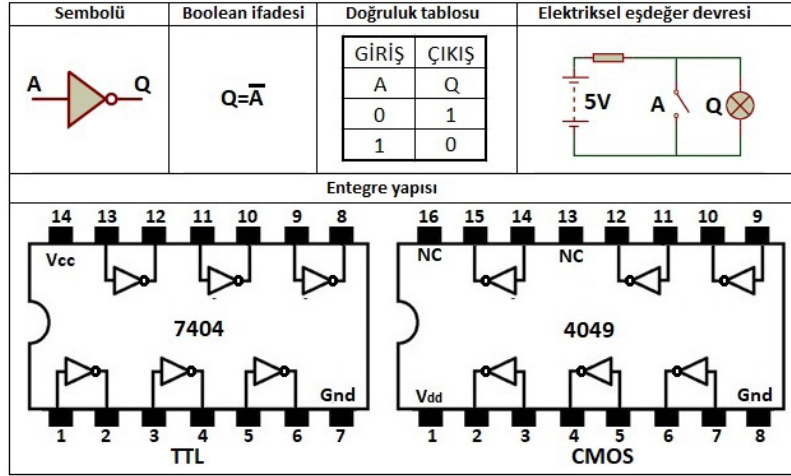
SIRA SİZDE

Uygulama 1.4'te verilen lojik devreyi 7432 entegresini kullanarak çizip çalıştırınız. Devrenin doğruluk tablosunu hazırlayınız. VEYA kapısı ve 4071 entegresi kullanarak hazırladığınız doğruluk tabloları ile 7432 entegresiyle hazırladığınız doğruluk tablosu arasında fark olup olmadığını tespit ediniz.

DEĞİL Kapısı (NOT Gate)

DEĞİL kapısı, bir adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişine uygulanan lojik bilgiyi, lojik çıkışa mantıksal değilini (tersini) alarak aktarır. Yani giriş lojik 1 ise çıkış lojik 0, giriş lojik 0 ise çıkış lojik 1 olur. DEĞİL kapısı evirici veya tersleyici olarak da isimlendirilir. Bolen ifadesi $Q = A$ (A DEĞİL) olarak ifade edilir.

Görsel 1.15'te DEĞİL kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.



Görsel 1.15: DEĞİL kapısı

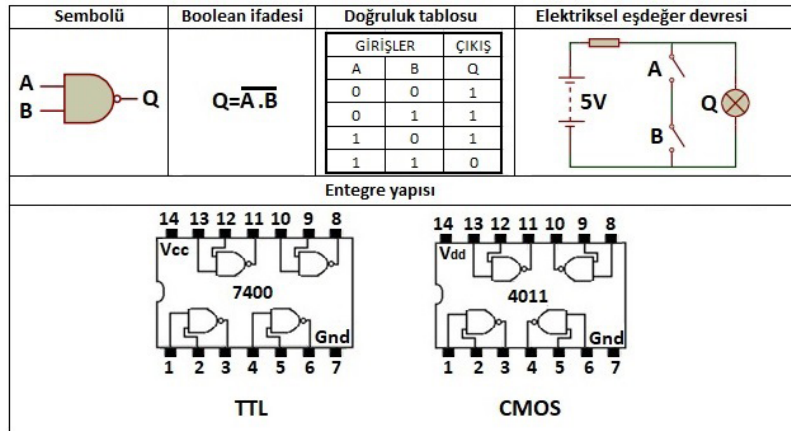
1.2.1.2. Diğer Lojik Kapılar

Temel lojik kapılar kullanılarak diğer lojik kapılar elde edilir. Bunlar sırasıyla şu başlıklar altında toplanır.

VE DEĞİL Kapısı (NAND Gate)

VE DEĞİL kapısı, genellikle iki adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişlerine uygulanan lojik bilgilerin mantıksal çarpımının değilini (tersini), lojik çıkışa aktarır. Girişlerin ikisi de lojik 1 ise çıkış lojik 0 olur. Diğer durumlarda çıkış lojik 1 olur. Bolen ifadesi $Q = \overline{A \cdot B}$ (A çarpı B'nin değili) olarak ifade edilir.

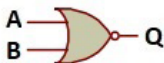
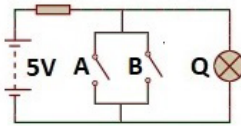
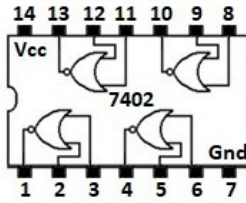
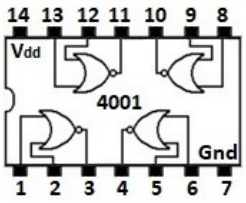
Görsel 1.16'da VE DEĞİL kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.



Görsel 1.16: VE DEĞİL kapısı

VEYA DEĞİL Kapısı (NOR Gate)

VEYA DEĞİL kapısı, genellikle iki adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişlerine uygulanan lojik bilgilerin mantıksal toplamının değerini (tersini), lojik çıkışa aktarır. Girişlerin ikisi de lojik 0 ise çıkış, lojik 1 olur. Diğer durumlarda çıkış lojik 0 olur. Bolen ifadesi $Q = \overline{A + B}$ (A toplam B'nin değili) olarak ifade edilir. Görsel 1.17'de VEYA DEĞİL kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.

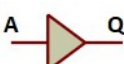
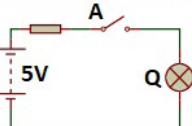
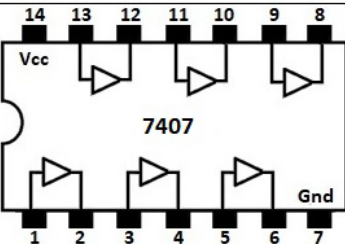
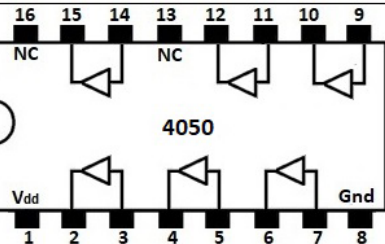
Sembolü	Boolean ifadesi	Doğruluk tablosu	Elektriksel eşdeğer devresi																		
	$Q = \overline{A + B}$	<table><tr><th colspan="2">GİRİŞLER</th><th>ÇIKIŞ</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	GİRİŞLER		ÇIKIŞ	A	B	Q	0	0	1	0	1	0	1	0	0	1	1	0	
GİRİŞLER		ÇIKIŞ																			
A	B	Q																			
0	0	1																			
0	1	0																			
1	0	0																			
1	1	0																			
Entegre yapısı																					
<div><div></div><div></div></div>																					
<div><div>TTL</div><div>CMOS</div></div>																					

Görsel 1.17: VEYA DEĞİL kapısı

TAMPON Kapısı (BUFFER Gate)

TAMPON kapısı, bir adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişine uygulanan lojik bilgiyi değiştirmeden lojik çıkışa aktarır. **TAMPON kapısı** elektronik devreler veya kullanılan diğer lojik kapılar arasında empedans uygunlaştırma görevi görür. Bolen ifadesi $Q = A$ 'dır.

Görsel 1.18'de TAMPON kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.


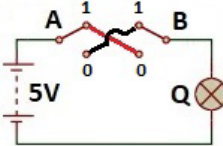
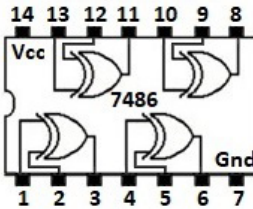
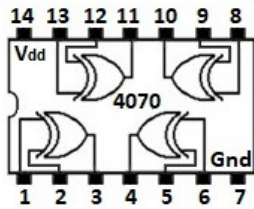
Sembolü	Boolean ifadesi	Doğruluk tablosu	Elektriksel eşdeğer devresi								
	$Q=A$	<table><tr><th>GİRİŞ</th><th>ÇIKIŞ</th></tr><tr><th>A</th><th>Q</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	GİRİŞ	ÇIKIŞ	A	Q	0	0	1	1	
GİRİŞ	ÇIKIŞ										
A	Q										
0	0										
1	1										
Entegre yapısı											
<div><div><p>TTL</p></div><div><p>CMOS</p></div></div>											

Görsel 1.18: TAMPON kapısı

ÖZEL VEYA Kapısı (EXOR Gate)

ÖZEL VEYA kapısı, genellikle iki adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişlerine uygulanan lojik bilgileri mantıksal karşılaştırır ve sonucu lojik çıkışa aktarır. Girişlerin her ikisi de aynı lojik değerde ise çıkış lojik 0 olur. Girişler birbirinden farklı lojik değerde ise çıkış lojik 1 olur. Bolen ifadesi $Q = A \oplus B$ (**A özel veya B**) olarak ifade edilir.

Görsel 1.19'da ÖZEL VEYA kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.

Sembolü	Boolean ifadesi	Doğruluk tablosu	Elektriksel eşdeğer devresi																		
	$Q = A \oplus B$ $Q = \bar{A}.B + A.\bar{B}$	<table><tr><th colspan="2">GİRİŞLER</th><th>ÇIKIŞ</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	GİRİŞLER		ÇIKIŞ	A	B	Q	0	0	0	0	1	1	1	0	1	1	1	0	
GİRİŞLER		ÇIKIŞ																			
A	B	Q																			
0	0	0																			
0	1	1																			
1	0	1																			
1	1	0																			
Entegre yapısı																					
<div><div><p>TTL</p></div><div><p>CMOS</p></div></div>																					

Görsel 1.19: ÖZEL VEYA kapısı



UYGULAMA

Adı:	ÖZEL VEYA Kapısı ile Mantıksal İşlemler	No.: 1.5
Amaç:	ÖZEL VEYA kapısı ile mantıksal işlemler yapmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda ÖZEL VEYA kapısı uygulama devresini elektronik çizim programında çizip çalıştırınız.

Kullanılacak Araç Gereç: Tablo 1.5'te belirtilmiştir.

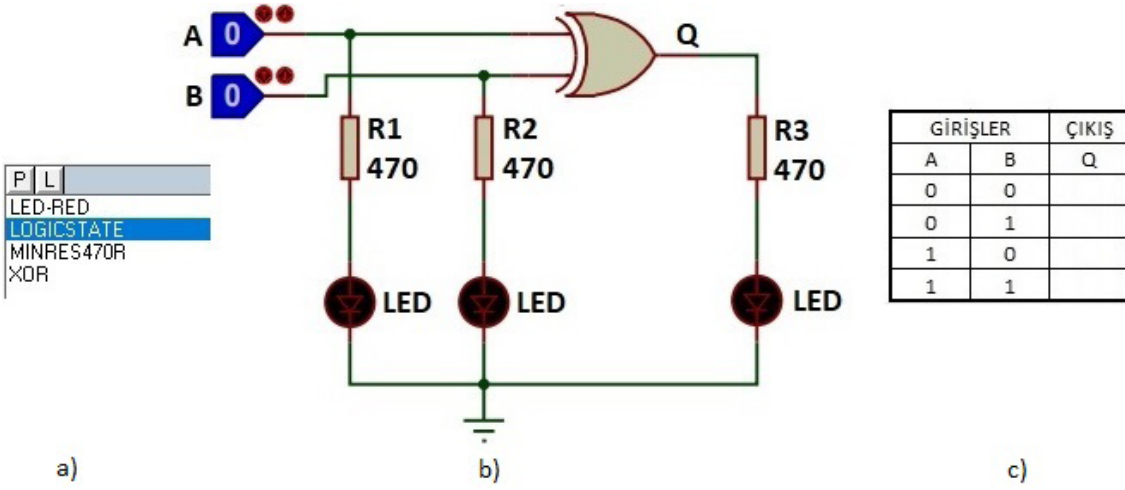
Tablo 1.5: Kullanılacak Araç Gereç Listesi

Adı	Özellği	Miktarı
Elektronik şema çizim programı	Windows tabanlı	1 adet

Uygulama Adımları:

1. Adım: Elektronik devre çizim programını çalıştırınız.

2. Adım: Görsel 1.20:b' de verilen ÖZEL VEYA kapısı uygulama devresini, Görsel 1.20:a' da gösterilen "DEVICES" penceresindeki devre elemanlarını kullanarak elektronik çizim programında çiziniz.



Görsel 1.20: a) Uygulama devre elemanları Görsel 1.20: b) Devre şeması Görsel 1.20: c) Doğruluk tablosu

3. Adım: A ve B lojik girişlerine, sırasıyla $(00)_2$, $(01)_2$, $(10)_2$ ve $(11)_2$ ikilik bilgilerini uygulayıp Q çıkışına bağlı LED'in ışık verdiği durumlara 1, ışık vermediği durumlara 0 yazarak Görsel 1.20:c' de verilen doğruluk tablosunu doldurunuz.

4. Adım: Doğruluk tablosuna yazdığınız değerleri öğretmeninize kontrol ettiriniz.

5. Adım: Çizim programı kütüphanesinden 4070 entegresi alarak devreyi tekrar çizip çalıştırınız ve adım 3'ü tekrarlayınız. XOR kapısı ile 4070 entegresinin doğruluk tablosunu karşılaştırınız.

6. Adım: Çalışmanız bitince bilgisayarı kapatınız.

7. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız diğer sayfadaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Elektronik çizim programı kütüphanesinden devrede kullanılacak elemanlar doğru seçildi.		
3. Elektronik çizim programında verilen devre doğru çizildi.		
4. Doğruluk tablosu doğru dolduruldu.		
5. Temizliğe dikkat edildi.		


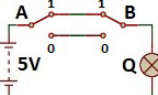
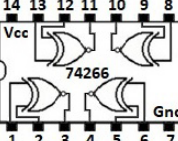
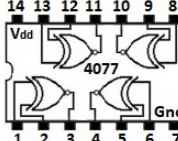


SIRA SİZDE

Uygulama 1.5'te verilen lojik devreyi 74LS386 entegresini kullanarak çizip çalıştırınız. Devrenin doğruluk tablosunu hazırlayınız. ÖZEL VEYA kapısı ve 4070 entegresi kullanarak hazırladığınız doğruluk tabloları ile 74LS386 entegresiyle hazırladığınız doğruluk tablosu arasında fark olup olmadığını tespit ediniz.

ÖZEL VEYA DEĞİL Kapısı (EXNOR Gate)

ÖZEL VEYA DEĞİL kapısı, genellikle iki adet girişi ve bir adet çıkışı bulunan lojik kapıdır. Girişlerine uygulanan lojik bilgileri mantıksal karşılaştırır ve sonucu lojik çıkışa aktarır. Girişlerin her ikisi de aynı lojik değerde ise çıkış lojik 1 olur. Girişler birbirinden farklı lojik değerde ise çıkış lojik 0 olur. Bolen ifadesi $Q = A \odot B$ (**A özel veya B'nin değili**) olarak ifade edilir. Görsel 1.21'de ÖZEL VEYA DEĞİL kapısının sembolü, bolen ifadesi, doğruluk tablosu, elektriksel eşdeğer devresi ve entegre yapısı görülmektedir.

Sembolü	Boolean ifadesi	Doğruluk tablosu	Elektriksel eşdeğer devresi																		
	$Q = A \oplus B$ $Q = A.B + \overline{A.B}$	<table><tr><th colspan="2">GİRİŞLER</th><th>ÇIKIŞ</th></tr><tr><th>A</th><th>B</th><th>Q</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	GİRİŞLER		ÇIKIŞ	A	B	Q	0	0	1	0	1	0	1	0	0	1	1	1	
GİRİŞLER		ÇIKIŞ																			
A	B	Q																			
0	0	1																			
0	1	0																			
1	0	0																			
1	1	1																			
Entegre yapısı																					
<div><div><p>74266</p><p>TTL</p></div><div><p>4077</p><p>CMOS</p></div></div>																					

Görsel 1.21: ÖZEL VEYA DEĞİL kapısı

1.2.2. Temel Lojik Kapılarla İşlemler

Lojik kapılar temel anlamda programlama devreleri olarak kullanılır. Lojik kapılarla mantıksal toplama (VEYA), çarpma (VE) ve tersini alma (tümleme mantık) gibi işlemler yapılır. Lojik kapılar ile devre tasarımı yapılırken giriş değişken sayısı önemlidir. Giriş sayısına bağlı olarak doğruluk tablosu hazırlanır. Lojik ifadeler, lojik kapılarla işleme alınmadan önce mümkün olduğu kadar sadeleştirilir. Sadeleştirme yapılırken bolen kanun ve kuralları, çift tersleme kuralı ve dö morgin (De Morgan) kuralı kullanılır. Sonuç olarak çarpımların toplamı veya toplamaların çarpımı şeklinde bolen ifadeler elde edilir. Elde edilen bolen ifadelerin lojik şemaları, lojik kapılarla çizilir. Bolen işlemleri şu şekildedir.

1.2.2.1. Bolen Çarpma

Bolen çarpma, VE işlemine eşdeğerdir. Kuralları şu şekildedir:

$$0 \cdot 0 = 0$$

$$0 \cdot 1 = 0$$

$$1 \cdot 0 = 0$$

$$1 \cdot 1 = 1$$



ÖRNEK 24

Aşağıda verilen bolen çarpma işlemini yapınız.

$$Q = (1 \cdot 0) \cdot (1 \cdot 1)$$

ÇÖZÜM: $Q = 0 \cdot 1$
 $Q = 0$



SIRA SİZDE

Aşağıda verilen bolen çarpma işlemini yapınız.

$$Q = (1 \cdot 1) \cdot (1 \cdot 0) \cdot (0 \cdot 0) \cdot (0 \cdot 1)$$

1.2.2.2. Bolen Toplama

Bolen toplama, VEYA işlemine eşdeğerdir. Kuralları şu şekildedir:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$



ÖRNEK 25

Aşağıda verilen bolen toplama işlemini yapınız.

$$Q = (1 + 0) + (1 + 1)$$

ÇÖZÜM: $Q = 1 + 1$
 $Q = 1$



SIRA SİZDE

Aşağıda verilen bolen toplama işlemini yapınız.

$$Q = (0 + 1) + (1 + 0) + (1 + 1)$$

1.2.2.3. Bolen Kanun ve Kuralları

George Boolean tarafından geliştirilen Bolen kanun ve kuralları, **VE**, **VEYA** ve **DEĞİL** temel mantıksal işlemlerinden oluşan sembolik bir sistemdir. Temel mantıksal işlemler kullanılarak toplama, çıkarma, çarpma, bölme ve karşılaştırma işlemleri yapılabilir. Bu işlemler temelde ikili işlemlerdir. Bu nedenle iki durumla açıklanabilir (Doğru-Yanlış, Açık-Kapalı, "1"-“0”). Bolen kanun ve kuralları lojik ifadelerin sadeleştirme işlemini gerçekleştirmek amacıyla kullanılır. Başlıca bolen kanun ve kuralları şunlardır:

VE (AND) Kanunu

1 ile VE işlemine alınan lojik ifade yine kendisine eşittir. 0 ile VE işlemine alınan lojik ifade 0'a eşittir.

$$A \cdot 1 = A$$

$$A \cdot 0 = 0$$

**ÖRNEK 26**

Aşağıda verilen lojik ifadeye VE kanununu uygulayınız.

$$Q = (A \cdot 0) \cdot (B \cdot 1)$$

ÇÖZÜM: $Q = 0 \cdot B$
 $Q = 0$

**SIRA SİZDE**

Aşağıda verilen lojik ifadeye VE kanununu uygulayınız.

$$Q = (A \cdot 1) \cdot (B \cdot 0) \cdot (C \cdot 0) \cdot (D \cdot 1)$$

VEYA (OR) Kanunu

1 ile VEYA işlemine alınan lojik ifade 1'e eşittir. 0 ile VEYA işlemine alınan lojik ifade yine kendisine eşittir.

$$A + 1 = 1$$

$$A + 0 = A$$

**ÖRNEK 27**

Aşağıda verilen lojik ifadeye VEYA kanununu uygulayınız.

$$Q = (A + 0) + (B + 1)$$

ÇÖZÜM: $Q = A + 1$
 $Q = 1$

**SIRA SİZDE**

Aşağıda verilen lojik ifadeye VEYA kanununu uygulayınız.

$$Q = (A + 1) + (B + 0) + (C + 1)$$

Özdeşlik Kanunu

Bir lojik ifadenin kendisi ile VE ya da VEYA işlemine alınması lojik ifadeyi değiştirmez.

$$A \cdot A = A$$

$$A + A = A$$

Değişme Kuralı

VE, VEYA işlemlerinde değişkenlerin yerlerinin değişmesi lojik ifadeyi değiştirmez.

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

Birleşme Kuralı

VE, VEYA işlemlerinde değişkenlerin kendi arasında birleşmesi lojik ifadeyi değiştirmez.

$$A + B + C = (A + B) + C = A + (B + C)$$

$$A \cdot B \cdot C = (A \cdot B) \cdot C = A \cdot (B \cdot C)$$

Dağılma Kuralı

VE, VEYA işlemlerinde çarpmanın toplama üzerine dağılması veya toplamın çarpma üzerine dağılması lojik ifadeyi değiştirmez.

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

Yutma Kuralı

Bir lojik değişken kendisinin de içinde bulunduğu VEYA işlemine alınmış bir lojik ifade ile VE işlemine alınırsa sonuç değişkenin kendisidir.

$$A \cdot (A + B) = A$$

Bir lojik değişken kendisinin de içinde bulunduğu VE işlemine alınmış bir lojik ifade ile VEYA işlemine alınırsa sonuç değişkenin kendisidir.

$$A + AB = A$$

Çift Tersleme Kuralı

Bir lojik ifadenin DEĞİL'inin DEĞİL'i yine kendisine eşittir.

$$\overline{\overline{A}} = A$$

$$\overline{(A + B)} = (\overline{A} \cdot \overline{B})$$

1.2.2.4. Dö Morgan (De Morgan) Teoremi

İngiliz matematikçi ve mantıkçı Augustus De Morgan (1806-1871) tarafından geliştirilmiştir. **De Morgan** teoremi, VE VEYA işlemine alınmış lojik değişken veya lojik ifadelerin kendi DEĞİL'leri arasındaki ilişkiyi gösteren bağıntıdır.

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

**ÖRNEK 28**

$Q = A \cdot (A + B)$ lojik ifadesini bolen kurallarını kullanarak sadeleştiriniz.

ÇÖZÜM: $Q = A \cdot (A + B)$
 $Q = AA + AB$
 $Q = A \cdot \underbrace{(1 + B)}_1$
 $Q = A$

**ÖRNEK 29**

$Q = (A+B) \cdot (A+C)$ lojik ifadesini bolen kurallarını kullanarak sadeleştiriniz.

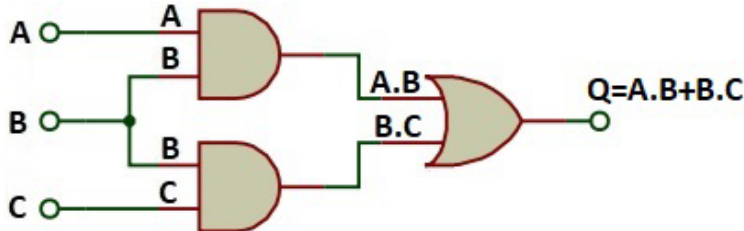
ÇÖZÜM: $Q = (A + B) \cdot (A + C)$
 $Q = AA + AC + AB + BC$
 $Q = A \cdot \underbrace{(1 + C + B)}_1 + BC$
 $Q = A + BC$

**SIRA SİZDE**

$Q = A(1 + B) + AD$ lojik ifadesini bolen kurallarını kullanarak sadeleştiriniz.

1.2.2.5. Çarpımların Toplamı (Minterm)

Çarpımların toplamı, VE kapısında çarpma işlemi yapılan lojik ifadelerin VEYA kapısında toplanmasıdır. $Q = A \cdot B + C \cdot D$, $Q = A \cdot B \cdot C + A \cdot C$ lojik ifadeleri çarpımların toplamını göstermektedir. $Q = A \cdot B + B \cdot C$ lojik ifadesini lojik kapı ile gösterilecek olursa önce VE kapılarında $A \cdot B$ ve $B \cdot C$ çarpma işlemi yapılır. Çarpma işlemi sonuçları VEYA kapısında toplanır (Görsel 1.22).



Görsel 1.22: $Q = A.B + B.C$ bolen ifadesinin kapılarla gösterilmesi

**ÖRNEK 30**

Aşağıda verilen çarpımların toplamı işlemini yapınız.

$$Q = (A \cdot 0) + (B \cdot 1) + (AB)$$

ÇÖZÜM: $Q = 0 + B + AB$

$$Q = B + AB$$

$$Q = B(1 + A)$$

$$Q = B \cdot 1$$

$$Q = B$$

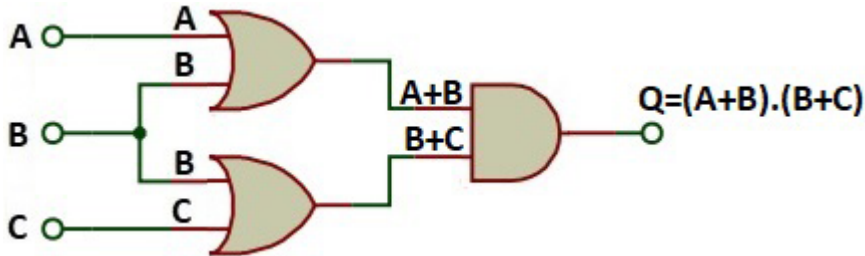
**SIRA SİZDE**

Aşağıda verilen çarpımların toplamı işlemini yapınız.

$$Q = (A \cdot 1) + (B \cdot 0) + (B \cdot C)$$

1.2.2.6. Toplamların Çarpımı (Maxterm)

VEYA kapısında toplama işlemi yapılan lojik ifadelerin VE kapısında çarpılmasıdır. $Q = (A + B) \cdot (C + D)$, $Q = (A + B + C) \cdot (A + C)$ lojik ifadeleri toplamaların çarpımını göstermektedir. $Q = (A + B) \cdot (B + C)$ lojik ifadesi lojik kapı ile gösterilecek olursa önce VEYA kapılarında $A + B$ sonra $B + C$ toplama işlemi yapılır. Toplama işlemi sonuçları VE kapısında çarpılır (Görsel 1.23).



Görsel 1.23: $Q = (A + B) \cdot (B + C)$ bolen ifadesinin kapılarla gösterilmesi

**ÖRNEK 31**

Aşağıda verilen toplamaların çarpımı işlemini yapınız.

$$Q = (A + 0) \cdot (B + 1) \cdot (A + B)$$

ÇÖZÜM: $Q = A \cdot 1 \cdot (A + B)$

$$Q = A \cdot (A + B)$$

$$Q = AA + AB$$

$$Q = A + AB$$

$$Q = A(1 + B)$$

$$Q = A$$



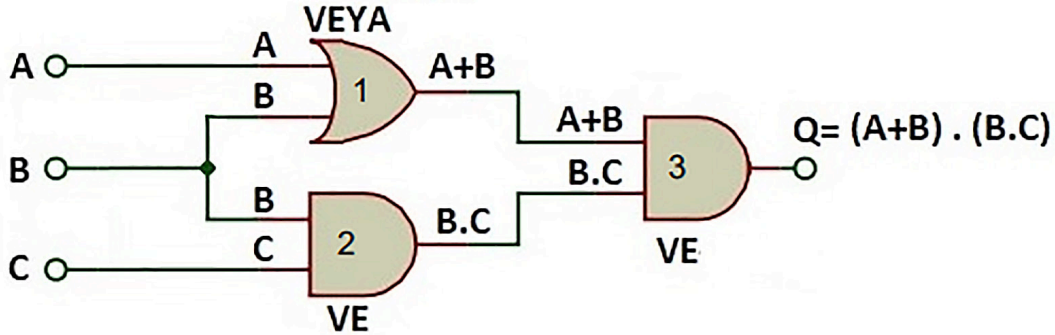
SIRA SİZDE

Aşağıda verilen toplamaların çarpımı işlemini yapınız.

$$Q = (A + 1) \cdot (B + 0) \cdot (A + B)$$

1.2.2.7. Lojik Devreden Lojik İfade Elde Etme

Lojik devrelerdeki Q çıkışı, en son kapıdan alınan lojik ifadedir. Doğruluk tablosu, Q lojik ifadesine göre doldurulur. Q lojik ifadesi bulunurken ilk önce girişte bulunan kapının lojik ifadesi bulunur. Daha sonra bağlandığı diğer kapı ya da kapılarla beraber lojik ifadeleri bulunur. En son çıkış kapısındaki lojik ifade bulunarak işlem tamamlanır. Bu işlemler yapılırken hata olmaması için her kapının çıkışına lojik ifadesi yazılmalıdır (Görsel 1.24).



Görsel 1.24: Lojik devrenin lojik ifadesi

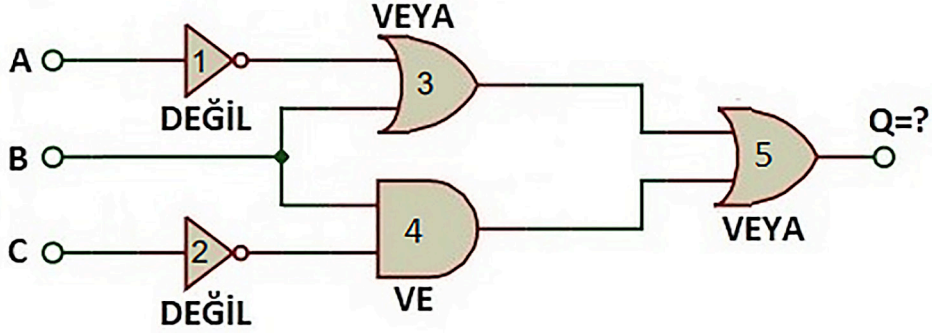
Görsel 1.24'teki lojik devre şemasından Q çıkış ifadesi şu şekilde bulunur:

1. A ve B değişkenleri 1 numaralı VEYA kapısının girişlerine uygulanmıştır. VEYA kapısının çıkış ifadesi $A + B$ olacaktır.
2. B ve C değişkenleri 2 numaralı VE kapısının girişlerine uygulanmıştır. VE kapısının çıkış ifadesi $B \cdot C$ olacaktır.
3. 1 numaralı VEYA kapısının çıkışı $A + B$ ile 2 numaralı VE kapının çıkışı $B \cdot C$, 3 numaralı VE kapısının girişlerine uygulanmıştır. VE kapısı çarpma işlemi yapacağından $Q = (A + B) \cdot (B \cdot C)$ olur.



ÖRNEK 32

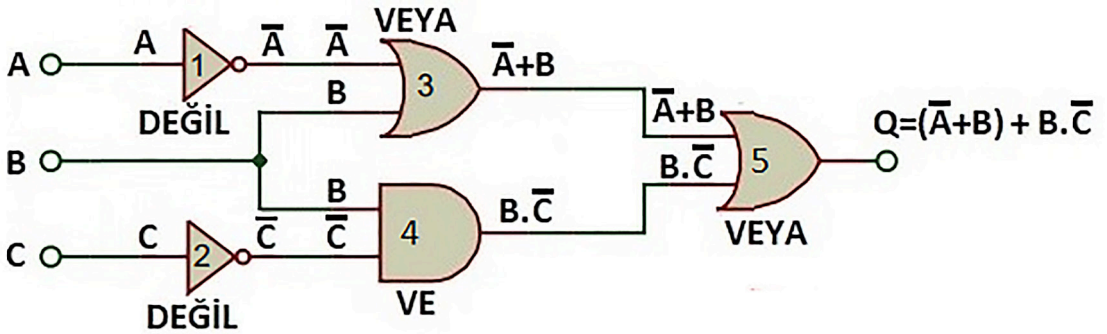
Görsel 1.25'te verilen lojik devrenin lojik ifadesini bulunuz.



Görsel 1.25: Örnek 32'ye ait lojik devre

ÇÖZÜM:

1. A değişkeni 1 numaralı DEĞİL kapısı girişine uygulanınca DEĞİL kapısının çıkışında \bar{A} ifadesini alacaktır. \bar{A} ve B değişkenleri 3 numaralı VEYA kapısının girişlerine uygulanmıştır. VEYA kapısının çıkış ifadesi $\bar{A} + B$ olacaktır.
2. C değişkeni 2 numaralı DEĞİL kapısına girişine uygulanınca DEĞİL kapısının çıkışında \bar{C} ifadesini alacaktır. B ve \bar{C} değişkenleri 4 numaralı VE kapısının girişlerine uygulanmıştır. VE kapısının çıkış ifadesi $B \cdot \bar{C}$ olacaktır.
3. 3 numaralı VEYA kapısının çıkışı $\bar{A} + B$ ile 4 numaralı VE kapının çıkışı $B \cdot \bar{C}$, 5 numaralı VEYA kapısının girişlerine uygulanmıştır. VEYA kapısı girişlerine uygulanan lojik ifadelerin mantıksal toplamını alacağından $Q = (\bar{A} + B) + B \cdot \bar{C}$ olacaktır. (Görsel 1.26)

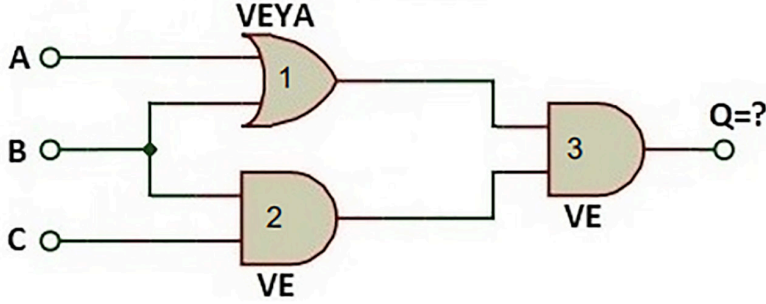


Görsel 1.26: Örnek 32'nin çözümü



SIRA SİZDE

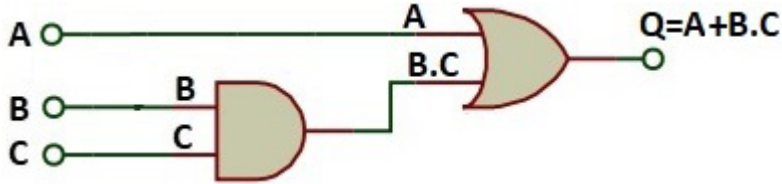
Görsel 1.27’de verilen lojik devrenin lojik ifadesini bulunuz.



Görsel 1.27: Lojik ifadesi bulunacak lojik devre

1.2.2.8. Lojik İfadeden Lojik Devre Çizimi

Lojik ifadeden lojik devre çizimi yapılırken öncelikle varsa parantez içinde olan ifadeler sonra sırası ile çarpma ve toplama ifadeleri uygun lojik kapı ile çizilir. Son olarak bulunan ifadeler toplam veya çarpım durumlarına göre uygun olan diğer kapılarla birleştirilir. $Q = A + B \cdot C$ ifadesinin lojik kapılarla çizilmiş devresi Görsel 1.28’de gösterilmektedir.



Görsel 1.28: $Q=A+B.C$ ifadesinin lojik devresi

Görsel 1.28’deki lojik devre şu işlem basamaklarına göre çizilir:

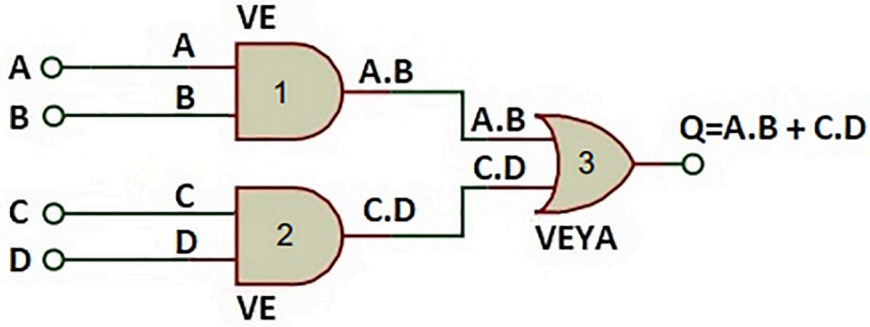
1. $Q = A + B \cdot C$ ifadesinde ilk önce $B \cdot C$ çarpım ifadesi VE kapısı ile çizilir.
2. A değişkeni $B \cdot C$ çarpım ifadesi ile toplandığından $B \cdot C$ ifadesine ait VE kapısının çıkışı ile A iki girişli VEYA kapısında toplanır.
3. VEYA kapısının çıkışı $Q = A + B \cdot C$ ifadesini verecektir.



ÖRNEK 33

$Q = (A \cdot B) + (C \cdot D)$ işlemini gerçekleştiren lojik devre şemasını çiziniz.

ÇÖZÜM: Yanıt Görsel 1.29'daki gibidir.



Görsel 1.29: Örnek 33'e ait lojik devre

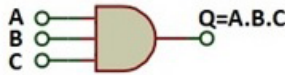


SIRA SİZDE

$Q = (A \cdot C) + (B \cdot D)$ işlemini gerçekleştiren lojik devre şemasını çiziniz.

1.2.2.9. Doğruluk Tablosu Hazırlama

Doğruluk tablosu, lojik devre tasarımında ve analizinde kullanılan çalışma durumu tablosudur. Doğruluk tablosu giriş değişkenlerinin alabileceği bütün durumlar için çıkış ifadesinin ne olduğunu gösterir. Lojik ifadede n sayıda giriş değişkeni varsa, bu değişkenler olası 2^n sayıda değişik durum alabilir. Örneğin, bir lojik devrenin iki ($n = 2$) giriş değişkeni varsa bu değişkenlerin alabileceği durum sayısı $2^2 = 4$ iken, üç giriş değişkeni ($n = 3$) için $2^3 = 8$, dört giriş değişkeni ($n = 4$) için $2^4 = 16$ farklı durum yazılır. Görsel 1.30'da üç girişli VE kapısının doğruluk tablosu görülmektedir.



	GİRİŞLER			ÇIKIŞ
	A	B	C	$Q = A.B.C$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Görsel 1.30: Üç girişli VE kapısının doğruluk tablosu

Görsel 1.30'da gösterilen doğruluk tablosu şu işlem basamaklarına göre hazırlanır:

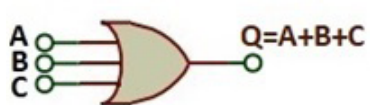
- 0 numaralı satırda $A = 0, B = 0, C = 0$ değerindedir. $Q_0 = A \cdot B \cdot C$ olduğundan $Q_0 = 0 \cdot 0 \cdot 0 = 0$ olur.
- 1 numaralı satırda $A = 0, B = 0, C = 1$ değerindedir. $Q_1 = A \cdot B \cdot C$ olduğundan $Q_1 = 0 \cdot 0 \cdot 1 = 0$ olur.

- 2 numaralı satırda $A = 0, B = 1, C = 0$ değerindedir. $Q_2 = A \cdot B \cdot C$ olduğundan $Q_2 = 0 \cdot 1 \cdot 0 = 0$ olur.
- 3 numaralı satırda $A = 0, B = 1, C = 1$ değerindedir. $Q_3 = A \cdot B \cdot C$ olduğundan $Q_3 = 0 \cdot 1 \cdot 1 = 0$ olur.
- 4 numaralı satırda $A = 1, B = 0, C = 0$ değerindedir. $Q_4 = A \cdot B \cdot C$ olduğundan $Q_4 = 1 \cdot 0 \cdot 0 = 0$ olur.
- 5 numaralı satırda $A = 1, B = 0, C = 1$ değerindedir. $Q_5 = A \cdot B \cdot C$ olduğundan $Q_5 = 1 \cdot 0 \cdot 1 = 0$ olur.
- 6 numaralı satırda $A = 1, B = 1, C = 0$ değerindedir. $Q_6 = A \cdot B \cdot C$ olduğundan $Q_6 = 1 \cdot 1 \cdot 0 = 0$ olur.
- 7 numaralı satırda $A = 1, B = 1, C = 1$ değerindedir. $Q_7 = A \cdot B \cdot C$ olduğundan $Q_7 = 1 \cdot 1 \cdot 1 = 1$ olur.

**ÖRNEK 34**

Üç girişli VEYA kapısının doğruluk tablosunu çiziniz.

ÇÖZÜM:



	GİRİŞLER			ÇIKIŞ
	A	B	C	$Q = A + B + C$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Görsel 1.31: Üç girişli VEYA kapısının doğruluk tablosu

Görsel 1.31’de gösterilen doğruluk tablosu şu işlem basamaklarına göre hazırlanır:

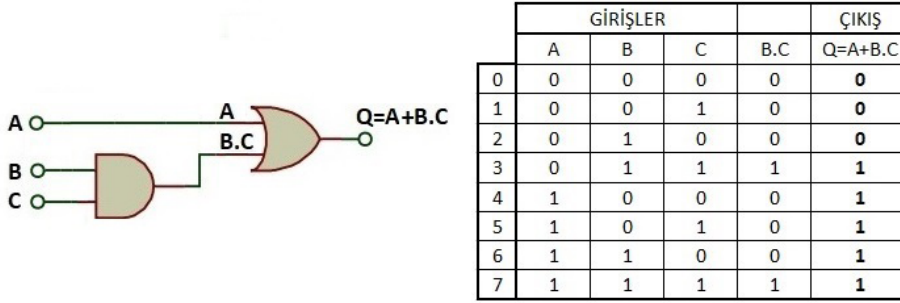
- 0 numaralı satırda $A = 0, B = 0, C = 0$ değerindedir. $Q_0 = A + B + C$ olduğundan $Q_0 = 0 + 0 + 0 = 0$ olur.
- 1 numaralı satırda $A = 0, B = 0, C = 1$ değerindedir. $Q_1 = A + B + C$ olduğundan $Q_1 = 0 + 0 + 1 = 1$ olur.
- 2 numaralı satırda $A = 0, B = 1, C = 0$ değerindedir. $Q_2 = A + B + C$ olduğundan $Q_2 = 0 + 1 + 0 = 1$ olur.
- 3 numaralı satırda $A = 0, B = 1, C = 1$ değerindedir. $Q_3 = A + B + C$ olduğundan $Q_3 = 0 + 1 + 1 = 1$ olur.
- 4 numaralı satırda $A = 1, B = 0, C = 0$ değerindedir. $Q_4 = A + B + C$ olduğundan $Q_4 = 1 + 0 + 0 = 1$ olur.
- 5 numaralı satırda $A = 1, B = 0, C = 1$ değerindedir. $Q_5 = A + B + C$ olduğundan $Q_5 = 1 + 0 + 1 = 1$ olur.
- 6 numaralı satırda $A = 1, B = 1, C = 0$ değerindedir. $Q_6 = A + B + C$ olduğundan $Q_6 = 1 + 1 + 0 = 1$ olur.
- 7 numaralı satırda $A = 1, B = 1, C = 1$ değerindedir. $Q_7 = A + B + C$ olduğundan $Q_7 = 1 + 1 + 1 = 1$ olur.



ÖRNEK 35

$Q = A + B \cdot C$ lojik ifadesinin doğruluk tablosunu çiziniz.

ÇÖZÜM:



Görsel 1.32: $Q=A+B.C$ lojik ifadesinin doğruluk tablosu

Görsel 1.32’de gösterilen doğruluk tablosu şu işlem basamaklarına göre hazırlanır:

- 0 numaralı satırda $A=0, B=0, C=0$ değerindedir. $Q_0 = A + B.C$ olduğundan $Q_0 = 0 + 0.0 = 0 + 0 = 0$ olur.
- 1 numaralı satırda $A=0, B=0, C=1$ değerindedir. $Q_1 = A + B.C$ olduğundan $Q_1 = 0 + 0.1 = 0 + 0 = 0$ olur.
- 2 numaralı satırda $A=0, B=1, C=0$ değerindedir. $Q_2 = A + B.C$ olduğundan $Q_2 = 0 + 1.0 = 0 + 0 = 0$ olur.
- 3 numaralı satırda $A=0, B=1, C=1$ değerindedir. $Q_3 = A + B.C$ olduğundan $Q_3 = 0 + 1.1 = 0 + 1 = 1$ olur.
- 4 numaralı satırda $A=1, B=0, C=0$ değerindedir. $Q_4 = A + B.C$ olduğundan $Q_4 = 1 + 0.0 = 1 + 0 = 1$ olur.
- 5 numaralı satırda $A=1, B=0, C=1$ değerindedir. $Q_5 = A + B.C$ olduğundan $Q_5 = 1 + 0.1 = 1 + 0 = 1$ olur.
- 6 numaralı satırda $A=1, B=1, C=0$ değerindedir. $Q_6 = A + B.C$ olduğundan $Q_6 = 1 + 1.0 = 1 + 0 = 1$ olur.
- 7 numaralı satırda $A=1, B=1, C=1$ değerindedir. $Q_7 = A + B.C$ olduğundan $Q_7 = 1 + 1.1 = 1 + 1 = 1$ olur.



UYGULAMA

Adı:	Lojik Devrenin Doğruluk Tablosunu Hazırlama	No.: 1.6
Amaç:	Temel lojik kapılarla mantıksal işlemler yapmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda verilen lojik devrenin doğruluk tablosunu elektronik çizim programında hazırlayınız.

Kullanılacak Araç Gereç: Tablo 1.6’da belirtilmiştir.

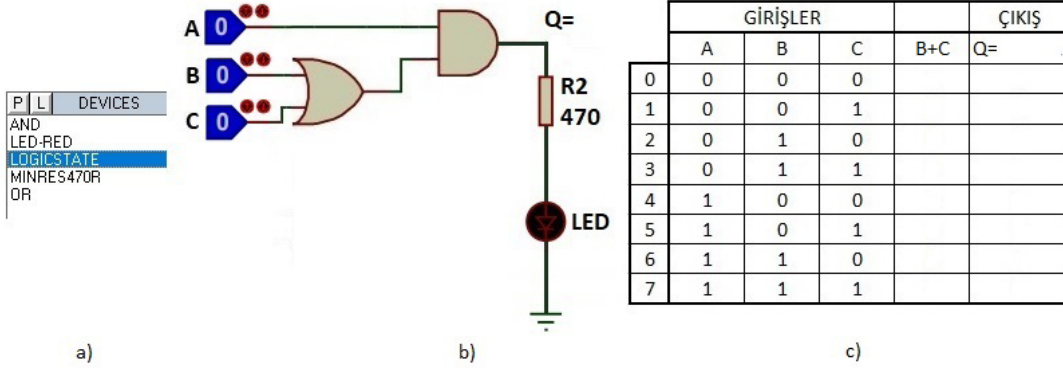
Tablo 1.6: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Elektronik şema çizim programı	Windows tabanlı	1 adet

Uygulama Adımları:

1. Adım: Elektronik devre çizim programını çalıştırınız.

2. Adım: Görsel 1.33:b’ de verilen lojik devreyi, Görsel 1.33:a’ da gösterilen “DEVICES” penceresindeki devre elemanlarını kullanarak elektronik çizim programında çiziniz.



Görsel 1.33: a) Uygulama devre elemanları Görsel 1.33: b) Devre şeması Görsel 1.33: c) Doğruluk tablosu

3. Adım: A, B ve C lojik girişlerine, sırasıyla $(000)_2$ ile $(111)_2$ arasındaki tüm ikilik bilgilerini uygulayıp Q çıkışına bağlı LED'in ışık verdiği durumlara 1, ışık vermediği durumlara 0 yazarak Görsel 1.33:c’ de verilen doğruluk tablosunu doldurunuz.

4. Adım: Doğruluk tablosuna yazdığınız değerleri öğretmeninize kontrol ettiriniz.

5. Adım: Çalışmanız bitince bilgisayarı kapatınız.

6. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Elektronik çizim programı kütüphanesinden devrede kullanılacak elemanlar doğru seçildi.		
3. Elektronik çizim programında verilen devre doğru çizildi.		
4. Doğruluk tablosu doğru dolduruldu.		
5. Temizliğe dikkat edildi.		



SIRA SİZDE

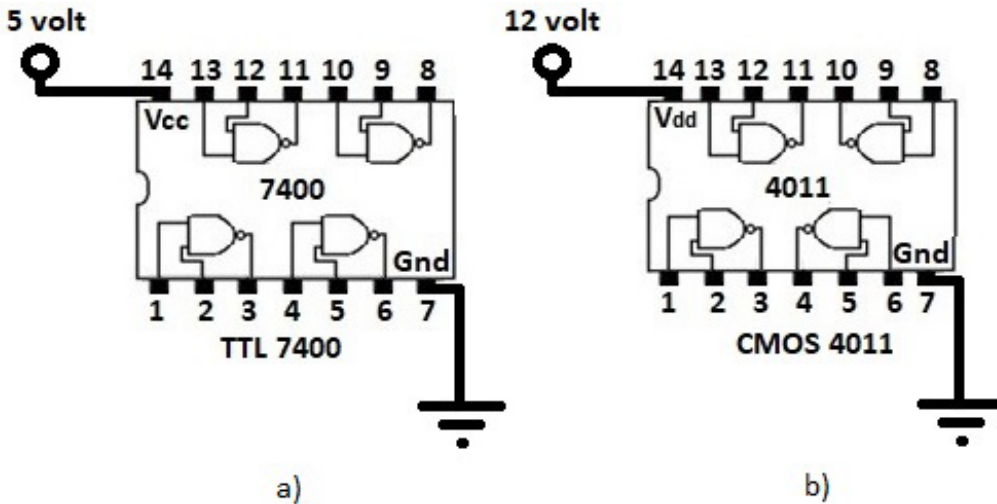
Uygulama 1.6'da verilen lojik devrede, VE kapısı ile VEYA kapısının yerini değiştirerek lojik devreyi tekrar çizip çalıştırınız. Devrenin doğruluk tablosunu hazırlayınız.

1.3. TEMEL LOJİK ENTEGRELERLE DEVRELER

Lojik devreler, entegre kullanılarak kurulurken **VE kapısı**, **VEYA kapısı** ve **DEĞİL kapısı** içeren temel lojik entegreler seçilir. Temel lojik entegreler dışında kapı kullanılacaksa ya diğer kapıları barındıran lojik entegreler kullanılır ya da temel lojik entegrelerden diğer kapılar elde edilir.

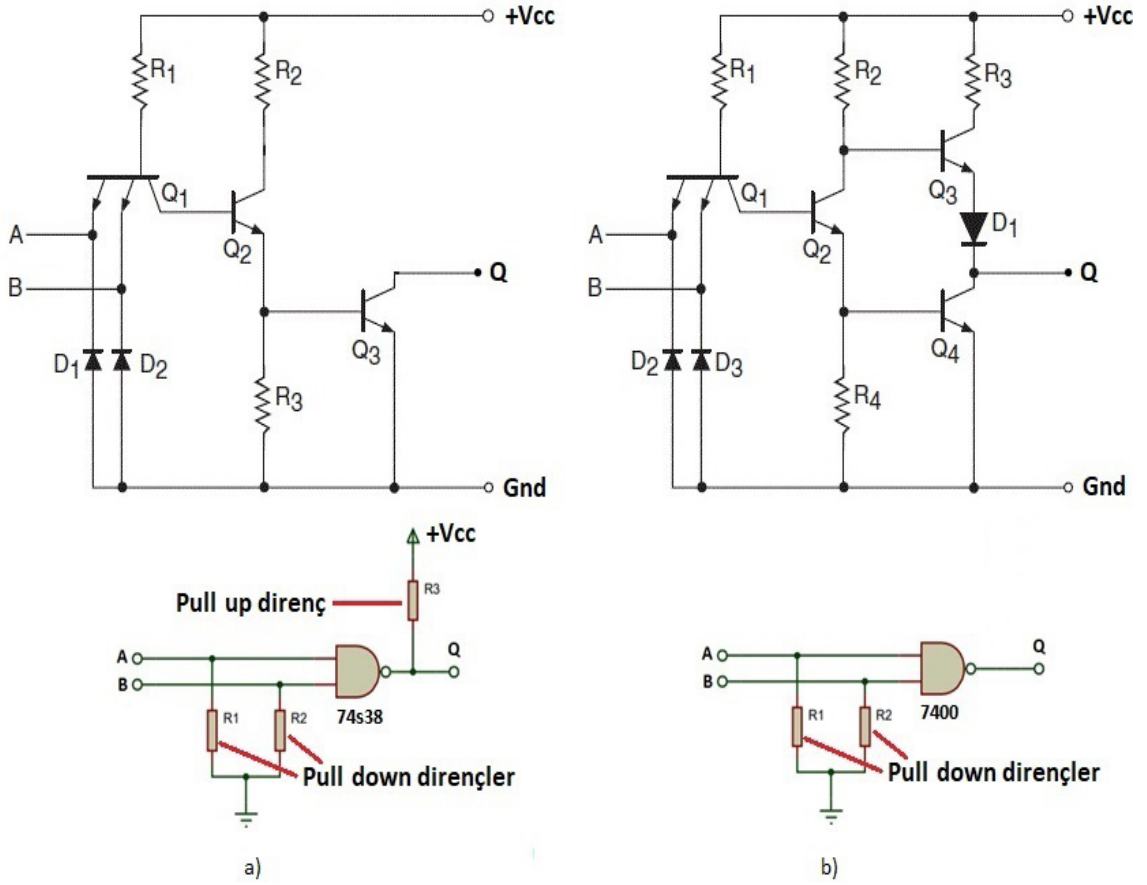
1.3.1. Temel Lojik Entegrelerin Seçimi

Lojik kapıların (elektronik devre olduğundan) besleme voltajları, entegre kataloğunda belirtilen voltaj değerlerinde olmalıdır. Lojik kapı, TTL teknolojisi kullanılarak üretilmiş bir entegre ise çalışma gerilimi 5 voltur. CMOS teknolojisi kullanılarak üretilmiş bir entegre ise çalışma gerilimi genellikle 12 voltur ancak 74HCxx gibi CMOS serisinde 5 volt kullanılabilir. Entegreler, katalogda belirtilen gerilim değerlerinin altındaki voltajlarda çalışmaz. Yüksek voltajda ise entegre zarar görür. Entegre içindeki tüm kapıların çalışması için besleme gerilimi istenilen gerilim seviyesinde olmalıdır. Görsel 1.34:a'da TTL ve Görsel 1.34:b'de CMOS entegrelerin besleme gerilimleri ve bağlantıları görülmektedir.



Görsel 1.34: a) TTL entegresinin besleme gerilimi bağlantısı Görsel 1.34: b) CMOS entegresinin besleme gerilimi bağlantısı

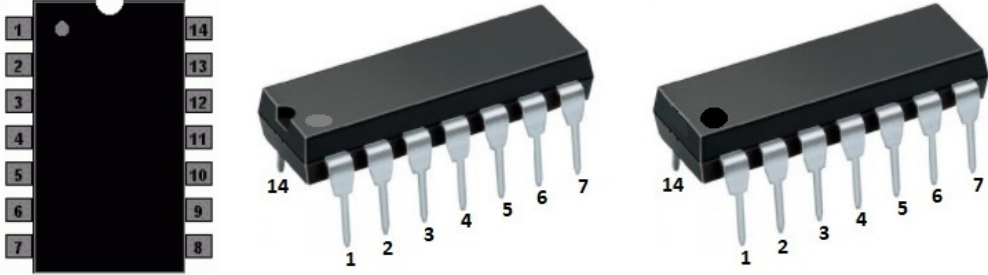
Mantıksal girişler genellikle direnç (pull down direnci) ile toprak voltajı seviyesinde tutularak lojik kapıların kararlı çalışması sağlanır. Lojik kapının çıkış katı boşa (open collector, open drain) olacak şekilde entegre tasarımı yapılmış ise muhakkak direnç (pull up direnci) yardımıyla besleme voltajına bağlanmalıdır. Görsel 1.35:a'da çıkış katı boşa olan 74s38 entegresinin iç yapısı ve içinde bulunan VE DEĞİL kapısının pull up bağlantısı gösterilmiştir. 7400 entegresinin yapısında da iki girişli VE DEĞİL kapıları olmasına rağmen çıkış katı boşa olmadığından pull up direncine gerek yoktur (Görsel 1.35:b).



Görsel 1.35: a) 74s38 iç yapısı ve pull down direnç bağlantısı Görsel 1.35: b) 7400 iç yapısı pull up ve pull down direnç bağlantısı

Entegre devreleri elektronik devrelere bağlarken entegrenin yönüne ve bacak numaralarına dikkat edilmesi gerekir. Entegre devreler çok çeşitli kılıflara sahiptir. Entegre devre üzerindeki yarım daire şeklindeki boşluğun solundaki bacak 1 olacak şekilde numaralandırılır. Bazı entegrelerde sadece sol üst köşede daire şeklinde baskı bulunur. Bu baskının solundaki bacak 1 olacak şekilde numaralandırılır.

Görsel 1.36’da on dört bacaklı entegrelerin bacak numaraları ve kılıfları gösterilmiştir. En çok kullanılan temel lojik entegreler 7404, 7408, 7432, 4069, 4071, 4081’dir.



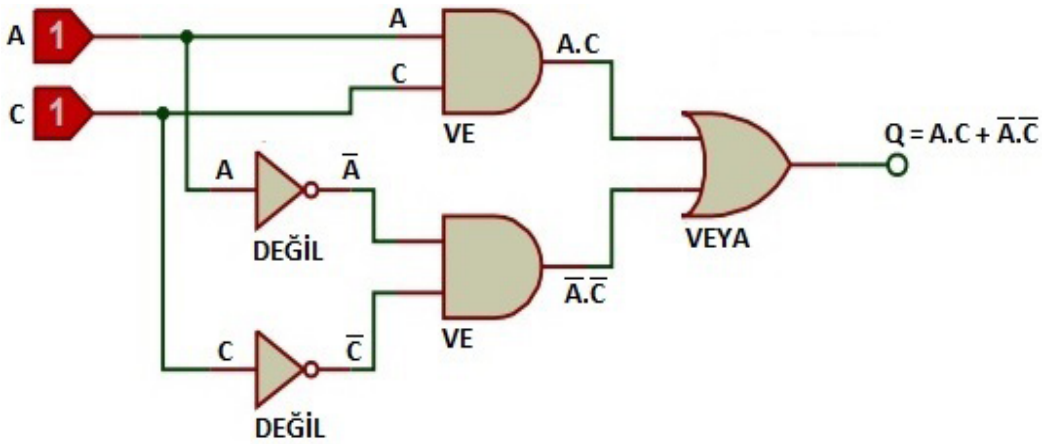
Görsel 1.36: Entegre bacak numaraları ve kılıfları



ÖRNEK 36

$Q = A.C + \bar{A}.\bar{C}$ lojik ifadesinde kullanılan lojik entegreleri belirleyiniz.

ÇÖZÜM: Görsel 1.37’ye bakıldığında lojik devrede \bar{A} , \bar{C} ifadeleri DEĞİL kapıları, $\bar{A}.\bar{C}$, $A.C$ ifadelerinden iki girişli VE kapıları, $\bar{A}.\bar{C} + A.C$ ifadeleri toplandığından iki girişli VEYA kapısı kullanılacağı anlaşılabacaktır. Entegre beslemesi 5 volt ise 7404 (DEĞİL), 7408 (VE), 7432 (VEYA) entegreleri kullanılacaktır. Entegre beslemesi 12 volt ise 4069 (DEĞİL), 4081 (VE), 4071 (VEYA) entegreleri seçilir.



Görsel 1.37: Örnek 36’ya ait lojik devre



SIRA SİZDE

$Q = (A . C) + (B . C)$ lojik ifadesinde kullanılacak lojik entegreleri seçiniz.



UYGULAMA

Adı:	$Q = (A + B) \cdot (C + D) + A \cdot C$ Lojik İfadesinin Entegrelerini Seçme	No.: 1.7
Amaç:	Temel lojik entegreleri seçmek.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda $Q = (A + B) \cdot (C + D) + A \cdot C$ lojik ifadesine ait lojik devreyi çizerek kullanılacak entegreleri yazınız.

Kullanılacak Araç Gereç: Tablo 1.7’de belirtilmiştir.

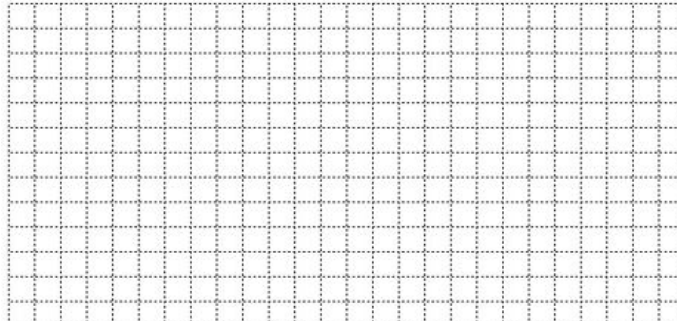
Tablo 1.7: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Elektronik şema çizim programı	Windows tabanlı	1 adet
Kalem	Kurşun	1 adet
Silgi	Kauçuk	1 adet

Uygulama Adımları:

1. Adım: Tablo 1.7’de belirtilen araç gereçleri hazırlayınız.

2. Adım: $Q = (A + B) \cdot (C + D) + A \cdot C$ lojik ifadesinin lojik devresini Görsel 1.38’de görülen çizim alanına çiziniz.



Görsel 1.38: Lojik devrenin çizileceği alan

3. Adım: Yaptığınız lojik devre çizimini öğretmeninize göstererek doğruluğunu onaylatınız.

4. Adım: Elektronik çizim programını çalıştırıp lojik devrede kullanılan kapılara uygun entegreleri bulup Tablo 1.8’e yazınız.

Tablo 1.8: Lojik Devrede Kullanılan Kapı Entegreleri

Lojik devrede kullanılan kapılar	TTL entegre karşılığı	CMOS entegre karşılığı
1.		
2.		

5. Adım: Tablo 1.8'i atölye öğretmeninize kontrol ettiriniz.

6. Adım: Çalışmanız bitince bilgisayarı kapatınız.

7. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Lojik devre doğru çizildi.		
3. Tablo 1.8'e entegreler doğru bulunup yazıldı.		
4. Temizliğe dikkat edildi.		



SIRA SİZDE

$Q = (A \cdot B) + (C \cdot D) \cdot (A + C)$ lojik ifadesinin lojik şemasını çiziniz. Lojik devrede kullanılacak entegreleri belirleyiniz.

1.3.2. Temel Lojik Devreler Kurma

Temel kapılar kullanılarak, basit lojik devrelerden karmaşık yapıdaki lojik devrelere kadar çok çeşitli devreler kurulabilir. Lojik devrede, Q çıkışı lojik ifadesi dikkate alınır. Q çıkışı lojik ifadesi en sade hâliyle verilirse doğrudan lojik devre kurulması aşamasına geçilir ancak Q çıkışı lojik ifadesi sadeleştirilmeden verilirse, lojik ifade bolen yasa ve kuralları ile veya karno (karnough) haritası yöntemi kullanılarak sadeleştirilir. Lojik ifadenin sadeleştirilmiş hâli uygun lojik entegreler seçilerek breadboard üzerine kurulur.

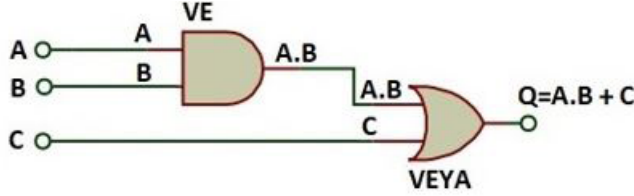
1.3.2.1. Sadeleştirilmiş Lojik İfadeden Devre Kurulması

Sadeleştirilmiş lojik ifadeye uygun lojik entegreler seçilerek breadboard üzerine lojik devre kurulur.

**ÖRNEK 37**

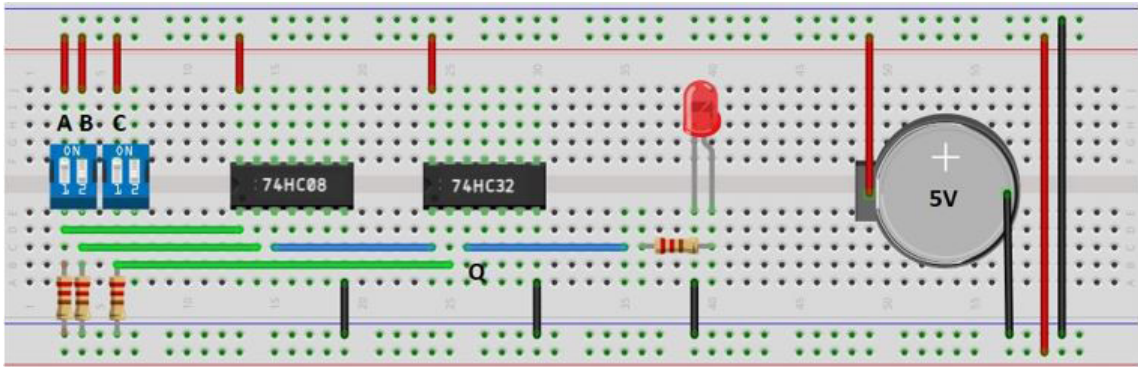
$Q = (A \cdot B) + C$ lojik ifadesine uygun lojik entegreleri belirleyip devresini kurunuz.

ÇÖZÜM: $Q = (A \cdot B) + C$ lojik ifadesinin lojik şeması Görsel 1.39'da çizilmiştir.



Görsel 1.39: $Q = AB + C$ lojik ifadesinin şeması

Görsel 1.39'da görüldüğü gibi lojik şemada VE ile VEYA kapısı kullanılmıştır. VE kapısı 7408 (TTL) veya 4081 (CMOS) entegresidir. VEYA kapısı 7432 (TTL) veya 4071 (CMOS) entegresidir. Entegrelerde besleme geriliminin artı ucu 14 No.lu bacağına, eksi ucu 7 No.lu bacağına bağlanır. Breadboard üzerine besleme, A, B ve C bağlantıları da yapılarak devre kurulumu tamamlanır (Görsel 1.40).



Görsel 1.40: $Q = AB + C$ lojik ifadesinin devresi

**UYGULAMA**

Adı:	$Q = (A + B) \cdot (C + D)$ Lojik İfadesinin Entegrelerini Seçerek Breadboard Üzerine Devre Kurulması	No.: 1.8
Amaç:	Temel lojik entegrelerle breadboard üzerine devre kurmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda $Q = (A + B) \cdot (C + D)$ lojik ifadesine ait entegreleri seçerek breadboard üzerine devreyi kurarak çalıştırınız.

Kullanılacak Araç Gereç: Tablo 1.9’da belirtilmiştir.

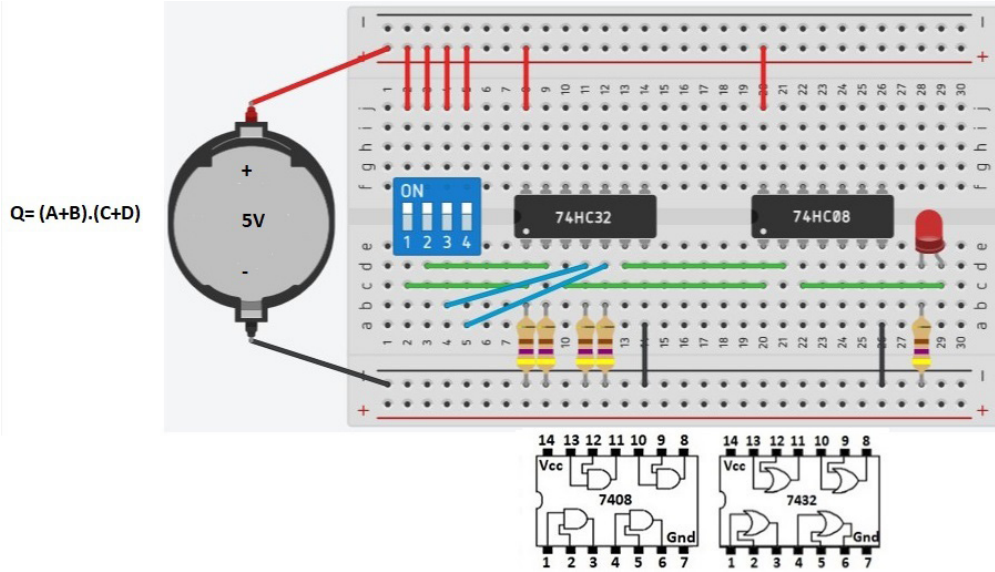
Tablo 1.9: Kullanılacak Araç Gereç Listesi

Adı	Özellği	Miktarı
Entegre	7408, 7432	2 Adet
Direnç	470Ω	5 Adet
DIP anahtar	Dörtlü	1 adet
LED	Kırmızı	1 adet
Breadboard		1 adet
Güç kaynağı	5 volt	1 adet
Kablo	Bağlantı kablosu	20 adet

Uygulama Adımları:

1. Adım: Tablo 1.9’da belirtilen malzemeleri hazırlayınız.

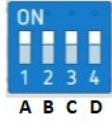
2. Adım: Görsel 1.41’de görüldüğü gibi devre elemanlarını ve bağlantı kablolarını breadboard üzerine yerleştiriniz.



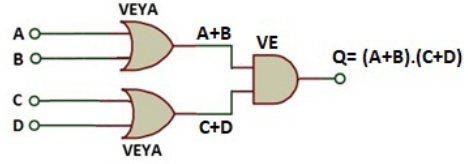
Görsel 1.41: Breadboard üzerine lojik devre kurulması

3. Adım: Öğretmeninizden onay aldıktan sonra güç kaynağını 5 volta ayarlayınız ve breadboard üzerine Görsel 1.41’de görüldüğü gibi bağlayınız.

4. Adım: DIP anahtarlardan A, B, C ve D lojik girişlerine Görsel 1.42’de gösterildiği gibi $(0000)_2$ ile $(1111)_2$ arası tüm lojik bilgileri girerek Q çıkış ifadesine göre, lojik devrenin doğruluk tablosunu hazırlayınız.



	GİRİŞLER				ÇIKIŞ
	A	B	C	D	$Q = (A+B).(C+D)$
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	



Görsel 1.42: $Q=(A+B).(C+D)$ lojik ifadesinin doğruluk tablosu

5. Adım: Hazırladığınız doğruluk tablosunu atölye öğretmeninize kontrol ettiriniz.

6. Adım: Çalışmanız bitince güç kaynağını kapatınız.

7. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Elektronik devre elemanları breadboard üzerine doğru yerleştirildi.		
3. Elektronik devre elemanları arasındaki bağlantılar doğru yapıldı.		
4. Lojik devrenin doğruluk tablosu doğru hazırlandı.		
5. Temizliğe dikkat edildi.		



SIRA SİZDE

$Q = (A + C) \cdot (B + D)$ lojik ifadesine ait entegreleri seçip breadboard üzerine devreyi kurarak çalıştırınız. Devrenin doğruluk tablosunu hazırlayınız.



UYGULAMA

Adı:	Q = AB + CD Lojik İfadesinin Entegrelerini Seçerek Breadboard Üzerine Devre Kurulması	No.: 1.9
Amaç:	Temel lojik entegrelerle breadboard üzerine devre kurmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda $Q = AB + CD$ lojik ifadesine ait entegreleri seçerek breadboard üzerine devreyi kurarak çalıştırınız.

Kullanılacak Araç Gereç: Tablo 1.10'da belirtilmiştir.

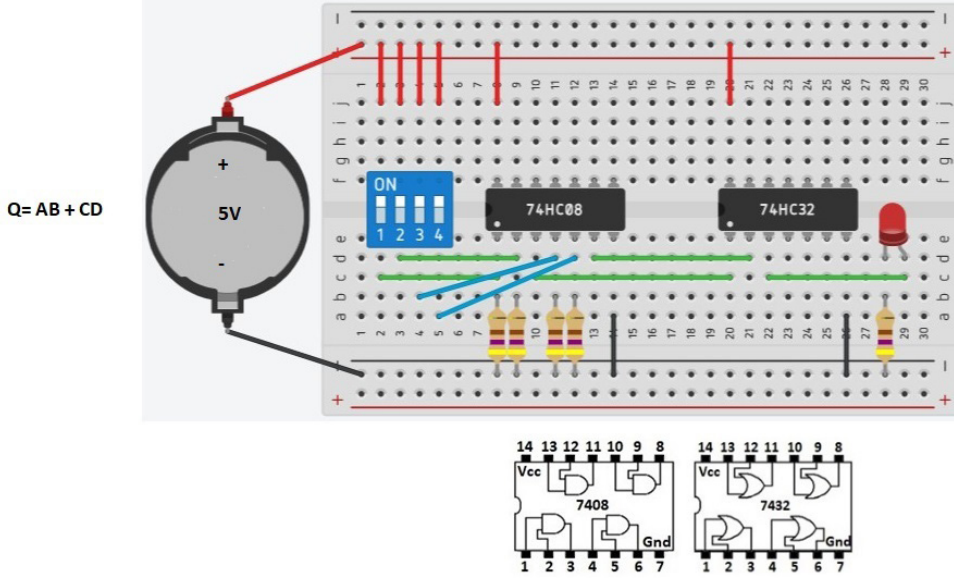
Tablo 1.10: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Entegre	7408, 7432	2 Adet
Direnç	470Ω	5 Adet
DIP anahtar	Dörtlü	1 adet
LED	Kırmızı	1 adet
Breadboard		1 adet
Güç kaynağı	5 volt	1 adet
Kablo	Bağlantı kablosu	20 adet

Uygulama Adımları:

1. Adım: Tablo 1.10'da belirtilen malzemeleri hazırlayınız.

2. Adım: Görsel 1.43'te görüldüğü gibi devre elemanlarını ve bağlantı kablolarını breadboard üzerine yerleştiriniz.

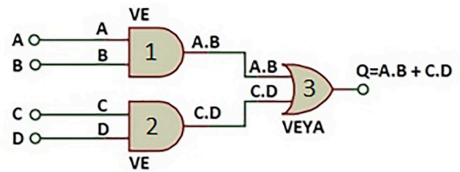


Görsel 1.43: Breadboard üzerine lojik devre kurulması

3. Adım: Öğretmeninden onay aldıktan sonra güç kaynağını 5 volta ayarlayınız ve breadboard üzerine Görsel 1.43'te görüldüğü gibi bağlayınız.

4. Adım: DIP anahtarlardan A, B, C ve D lojik girişlerine Görsel 1.44'te gösterildiği gibi $(0000)_2$ ile $(1111)_2$ arası tüm lojik bilgileri girerek Q çıkış ifadesine göre, lojik devrenin doğruluk tablosunu hazırlayınız.

	GİRİŞLER				ÇIKIŞ
	A	B	C	D	$Q = AB + CD$
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	



Görsel 1.44: $Q=AB+CD$ lojik ifadesinin doğruluk tablosu

5. Adım: Hazırladığınız doğruluk tablosunu atölye öğretmeninize kontrol ettiriniz.

6. Adım: Çalışmanız bitince güç kaynağını kapatınız.

7. Adım: Çalışma ortamını temizleyiniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Elektronik devre elemanları breadboard üzerine doğru yerleştirildi.		
3. Elektronik devre elemanları arasındaki bağlantılar doğru yapıldı.		
4. Lojik devrenin doğruluk tablosu doğru hazırlandı.		
5. Temizliğe dikkat edildi.		



SIRA SİZDE

$Q = (A \cdot C) + (B \cdot D)$ lojik ifadesine ait entegreleri seçip breadboard üzerine devreyi kurarak çalıştırınız. Devrenin doğruluk tablosunu hazırlayınız.

1.3.2.2. Karno (Karnaugh) Haritası Kullanılarak Devre Kurma

Karno haritası, bolen ifadeleri sadeleştirmek için kullanılan grafiksel bir yöntemdir. **Maurice Karnaugh** tarafından 1953'te geliştirilmiştir. Karno haritası, giriş değişkeni sayısına bağlı olarak hücrelerden oluşur. İkili sistemde işlem yapıldığından $n=\text{giriş değişkeni}$ sayısından 2^n formülüyle hücre sayısı belirlenir. Karno haritasında amaç 1'leri en çok grup hâlinde toplamaktır. Gruplama 1, 2, 4, 8, 16 gibi ikinin katları şeklinde alınır. Grup ne kadar büyük ise lojik ifade o kadar sadedir. Her bir grup kendi lojik ifadesine sahiptir ve VE kapısı ile gösterilir. Q çıkış lojik ifadesi ise tüm grupların toplanması ile elde edilir ve VEYA kapısı ile gösterilir.

İki Değişkenli Karno Haritası

Giriş değişkenleri A ve B iki adet olduğundan karno haritası Görsel 1.45:b'deki gibi dört hücreli çizilir (2^2). Görsel 1.45:a'daki doğruluk tablosunda Q çıkış ifadesi altında gösterilen m0, m1, m2, m3 karşılık gelen değerler karno haritasında gösterilen kısma yerleştirilir. Doğruluk tablosunda gösterilen m0, m1, m2 ve m3'e ait lojik ifadeleri Görsel 1.45:c'de gösterilmiştir.

GİRİŞLER ÇIKIŞ			
	A	B	Q
0	0	0	m0
1	0	1	m1
2	1	0	m2
3	1	1	m3

a)

		B	
Q	A	0	1
	0	m0 ①	m1 ②
	1	m2 ③	m3 ④

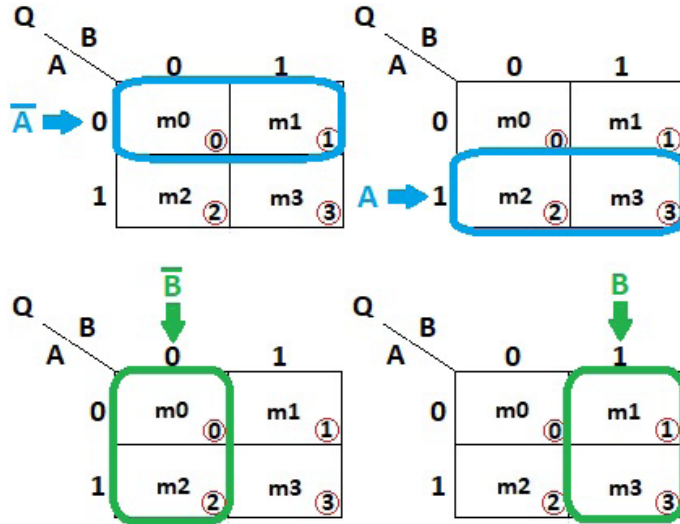
b)

c)

$m0 = \bar{A}\bar{B}$
 $m1 = \bar{A}B$
 $m2 = A\bar{B}$
 $m3 = AB$

Görsel 1.45: a) Doğruluk tablosu Görsel 1.45: b) Karno haritası Görsel 1.45: c) Lojik ifadeler

Karno haritasında satırlar A değişkenini gösterir. A değişkeninin 0 olduğu satır A'nın tersi, 1 olduğu satır A'nın kendisidir. Sütunlar B değişkenini gösterir. B değişkeninin 0 olduğu sütun B'nin tersi, 1 olduğu sütun B'nin kendisidir (Görsel 1.46).



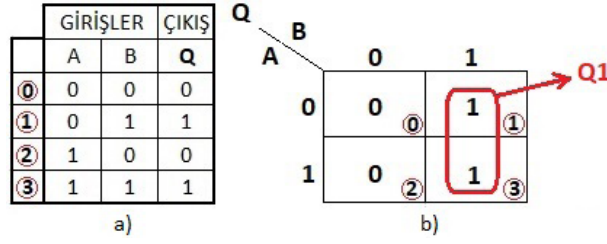
Görsel 1.46: A ve B değişkenlerinin karno haritası üzerinde gösterilmesi



ÖRNEK 38

$Q = \bar{A} \cdot B + A \cdot B$ lojik ifadesini karno haritası kullanarak sadeleştiriniz.

ÇÖZÜM:



Görsel 1.47: a) Örnek 38'in doğruluk tablosu Görsel 1.47: b) karno haritası

$Q = \bar{A} \cdot B + A \cdot B$ lojik çıkış ifadesine göre Görsel 1.47:a'daki doğruluk tablosu hazırlanır. $\bar{A} \cdot B$ ifadesi karno haritasının 1.hücresinin 1 olduğunu belirtiyor. $A \cdot B$ ifadesi karno haritasının 3.hücresinin 1 olduğunu belirtiyor. Diğer hücreler kullanılmadığından 0 olarak yazılır.

Karno haritasında 1'lerin oluşturacağı grupların en büyüğünü işaretlemek esas olduğundan Görsel 1.47:b'deki Q1 ifadesi gruplandırılır. Grupların 1, 2, 4, 8, 16 gibi ikinin katları olacağı unutulmamalıdır.

Q1 ifadesinde A ve B değişkenlerinin olup olmadığına bakılır. Q1 ifadesinde A hem 0 hem 1 değerini aldığından A değişkeni yazılmaz. Q1 ifadesinde B olduğu için yazılır. Böylece **Q1 = B** olur. $Q = Q1$ olduğundan **Q = B** ifadesi olarak bulunur.

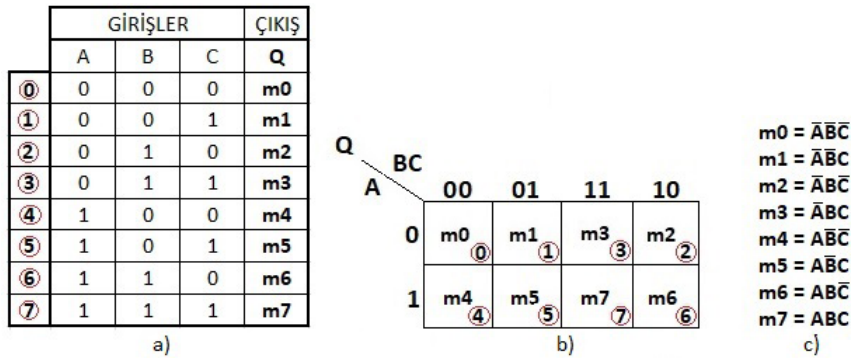


SIRA SİZDE

$Q = A \cdot \bar{B} + A \cdot B$ lojik ifadesini karno haritası kullanarak sadeleştiriniz.

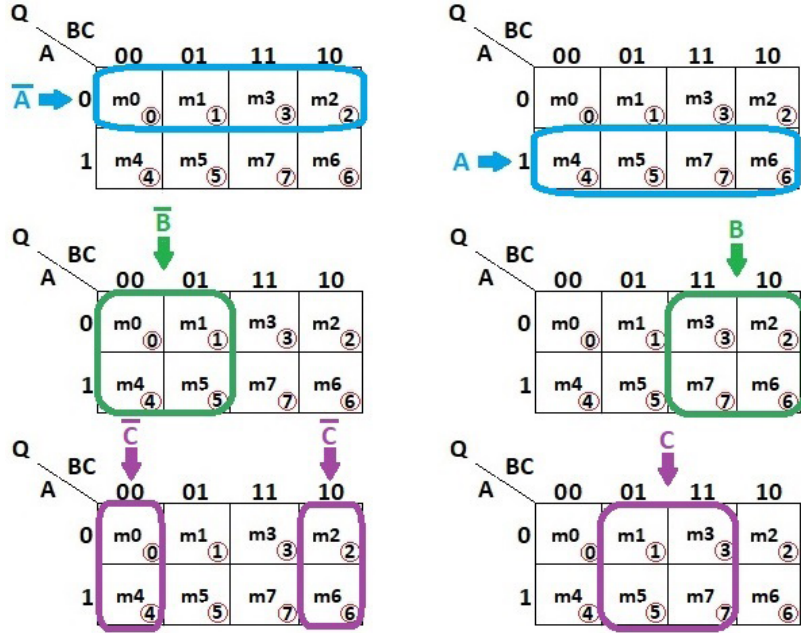
Üç Değişkenli Karno Haritası

Giriş değişkenleri A, B ve C üç adet olduğundan, karno haritası Görsel 1.48:b'deki gibi sekiz hücreli çizilir (2^3). Görsel 1.48:a'daki doğruluk tablosunda Q çıkış ifadesi altında gösterilen m0, m1, m2, m3, m4, m5, m6, m7'ye karşılık gelen değerler karno haritasındaki gösterilen kısma yerleştirilir. Doğruluk tablosunda gösterilen m0, m1, m2, m3, m4, m5, m6, m7'ye ait lojik ifadeleri Görsel 1.48:c'de gösterilmiştir.



Görsel 1.48: a) Doğruluk tablosu Görsel 1.48: b) Karno haritası Görsel 1.48: c) Lojik ifadeler

Karno haritasında satırlar A değişkenini gösterir. A değişkeninin 0 olduğu satır A'nın tersi, 1 olduğu satır A'nın kendisidir. Sütunlar B ve C değişkenlerini gösterir. B değişkeninin 00 ve 01 olduğu sütun B'nin tersi, 11 ve 10 olduğu sütun B'nin kendisini gösterirken C değişkeninin 00 ve 10 olduğu sütun C'nin tersi, 01 ve 11 olduğu sütun C'nin kendisini gösterir (Görsel 1.49).



Görsel 1.49: A, B ve C değişkenlerinin karno haritası üzerinde gösterilmesi



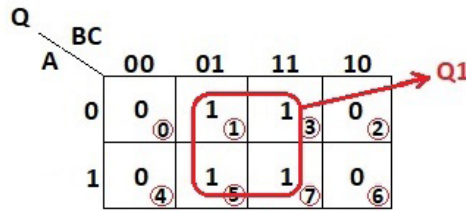
ÖRNEK 39

$Q = A \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C} + A \cdot \bar{B} \cdot C + A \cdot B \cdot C$ lojik ifadesini karno haritası kullanarak sadeleştiriniz.

ÇÖZÜM:

	GİRİŞLER			ÇIKIŞ
	A	B	C	Q
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

a)



b)

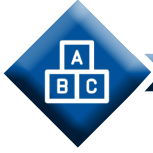
Görsel 1.50: a) Örnek 39'un doğruluk tablosu Görsel 1.50: b) karno haritası

$Q = A \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C} + A \cdot \bar{B} \cdot C + A \cdot B \cdot C$ çıkış ifadesine göre Görsel 1.50:a'daki doğruluk tablosu hazırlanır. $\bar{A} \cdot \bar{B} \cdot C$ ifadesi karno haritasının 1.hücresinin 1 olduğunu belirtiyor. $\bar{A} \cdot B \cdot C$ ifadesi karno haritasının 3.hücresinin 1 olduğunu belirtiyor. $A \cdot \bar{B} \cdot C$ ifadesi karno haritasının 5.hücresinin 1

olduğunu belirtiyor. A.B.C ifadesi karno haritasının 7.hücresinin 1 olduğunu belirtiyor. Diğer hücreler kullanılmadığından 0 olarak yazılır.

Karno haritasında 1'lerin oluşturacağı grupların en büyüğünü işaretlemek esas olduğundan Görsel 1.50:b'deki Q1 ifadesi gruplandırılır. Grupların 1, 2, 4, 8, 16 gibi ikinin katları olacağı unutulmamalıdır.

Q1 ifadesinde A, B ve C değişkenlerinin olup olmadığına bakılır. Q1 ifadesinde A hem 0 hem 1 değerini aldığından A değişkeni yazılmaz. Q1 ifadesinde B hem 0 hem 1 değerini aldığından B değişkeni yazılmaz. Q1 ifadesinde C olduğu için yazılır. Böylece **Q1 = C** olur. Q = Q1 olduğundan **Q=C** ifadesi olarak bulunur.



UYGULAMA

Adı:	Üç Değişkenli Karno Haritası ile Bulunan Lojik İfadeye Ait Devrenin Breadboard Üzerine Kurulması	No.: 1.10
Amaç:	Temel lojik entegrelerle breadboard üzerine devre kurmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda $\overline{Q} = \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot B \cdot C + A \cdot B \cdot \overline{C}$ lojik ifadesini karno haritası ile sadeleştirip, breadboard üzerine kurunuz.

Kullanılacak Araç Gereç: Tablo 1.11'de belirtilmiştir.

Tablo 1.11: Kullanılacak Araç Gereç Listesi

Adı	Özellği	Miktarı
Entegre	7408, 7432, 7404	3 Adet
Direnç	470Ω	4 Adet
DIP anahtar	Dörtlü	1 adet
LED	Kırmızı	1 adet
Breadboard		1 adet
Güç kaynağı	5 volt	1 adet
Kablo	Bağlantı kablosu	20 adet

Uygulama Adımları:

1. Adım: Tablo 1.11'de belirtilen malzemeleri hazırlayınız.

2. Adım: Verilen $Q = \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C}$ lojik ifadesine göre Görsel 1.51:a'daki doğruluk tablosunun Q çıkış değerlerini hesaplayıp doldurunuz.

	GİRİŞLER			ÇIKIŞ
	A	B	C	Q
①	0	0	0	$\bar{A}\bar{B}\bar{C}$ (.....)
②	0	0	1	$\bar{A}\bar{B}C$ (.....)
③	0	1	0	$\bar{A}B\bar{C}$ (.....)
④	0	1	1	$\bar{A}BC$ (.....)
⑤	1	0	0	$A\bar{B}\bar{C}$ (.....)
⑥	1	0	1	$A\bar{B}C$ (.....)
⑦	1	1	0	$AB\bar{C}$ (.....)
⑧	1	1	1	ABC (.....)

a)

		BC			
A	Q	00	01	11	10
	0	①	②	③	④
1		⑤	⑥	⑦	⑧

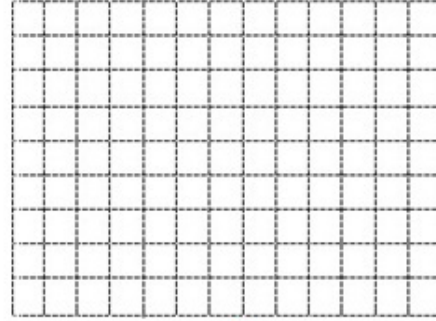
b)

Q1 =

Q2=

Q=

c)



d)

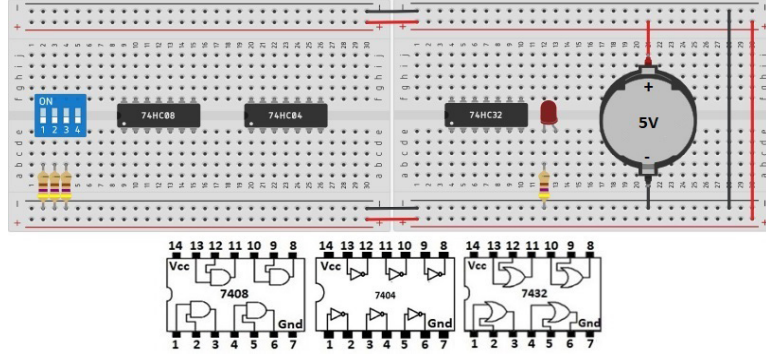
Görsel 1.51: a) Uygulama 1.10'a ait doğruluk tablosu Görsel 1.51: b) Karno haritası Görsel 1.51: c) Lojik ifade Görsel 1.51: d) Lojik diyagram

3. Adım: Doğruluk tablosunda 1'leri dikkati alarak Görsel 1.51:b'deki karno haritasını doldurunuz. Karno haritasına 1'leri doğru yerleştirdiyseniz Q1 ve Q2 olmak üzere iki grup bulacaksınız.

4. Adım: Karno haritasında grupları sadeleştirerek Q1, Q2 ve $Q = Q1 + Q2$ olarak bulduğunuz lojik ifadeleri 1.51:c'de verilen bölüme yazınız.

5. Adım: Bulduğunuz lojik ifadenin lojik şemasını 1.51:d'de verilen kısma çiziniz.

6. Adım: Çalışma ortamında dikkati dağıtacak malzemeleri bulundurmamaya dikkat ediniz. İş güvenliği tedbirlerini alarak Görsel 1.52’de görüldüğü gibi devre elemanlarını breadboard üzerine yerleştiriniz.



Görsel 1.52: Breadboard üzerine kurulacak uygulama devresi

7. Adım: Bağlantı kablolarını lojik şemaya uygun olarak breadboard üzerine yerleştiriniz.

8. Adım: Öğretmeninden onay aldıktan sonra güç kaynağını 5 volta ayarlayınız ve breadboard üzerine Görsel 1.52’de görüldüğü gibi bağlayınız.

9. Adım: A, B, C girişlerini Görsel 1.51:a’da gösterildiği gibi devreye uygulayarak, lojik devrenin doğruluk tablosunun doğru hazırlanıp hazırlanmadığını kontrol ediniz.

10. Adım: Elde ettiğiniz sonuçları öğretmeninize kontrol ettiriniz.

11. Adım: Çalışmanız bitince güç kaynağını kapatınız.

12. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmanız aşağıdaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Doğruluk tablosu doğru hazırlandı.		
3. Karno haritası doğru yerleştirildi.		
4. Sadeleştirme doğru yapıldı.		
5. Breadboard üzerine lojik devre doğru kuruldu.		
6. Breadboard üzerine kurulan devre, doğruluk tablosundaki Q çıkış değerlerine uygun çalıştırıldı.		
7. Temizliğe dikkat edildi.		



SIRA SİZDE

$Q = A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C}$ lojik ifadesini karno haritası kullanarak sadeleştiriniz. Lojik devresini breadboard üzerine kurunuz.

Dört Değişkenli Karno Haritası

Giriş değişkenleri A, B, C ve D dört adet olduğundan, karno haritası Görsel 1.53:b'deki gibi onaltı hücreli çizilir (2^4). Görsel 1.53:a'daki doğruluk tablosunda Q çıkış ifadesi altında gösterilen m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15'e karşılık gelen değerler karno haritasındaki gösterilen kısma yerleştirilir. Doğruluk tablosunda gösterilen m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15'e ait lojik ifadeleri Görsel 1.53:c'de gösterilmiştir.

	GİRİŞLER				ÇIKIŞ
	A	B	C	D	
0	0	0	0	0	m0
1	0	0	0	1	m1
2	0	0	1	0	m2
3	0	0	1	1	m3
4	0	1	0	0	m4
5	0	1	0	1	m5
6	0	1	1	0	m6
7	0	1	1	1	m7
8	1	0	0	0	m8
9	1	0	0	1	m9
10	1	0	1	0	m10
11	1	0	1	1	m11
12	1	1	0	0	m12
13	1	1	0	1	m13
14	1	1	1	0	m14
15	1	1	1	1	m15

a)

		CD			
		00	01	11	10
AB	00	m0 0	m1 1	m3 3	m2 2
	01	m4 4	m5 5	m7 7	m6 6
	11	m12 12	m13 13	m15 15	m14 14
	10	m8 8	m9 9	m11 11	m10 10

b)

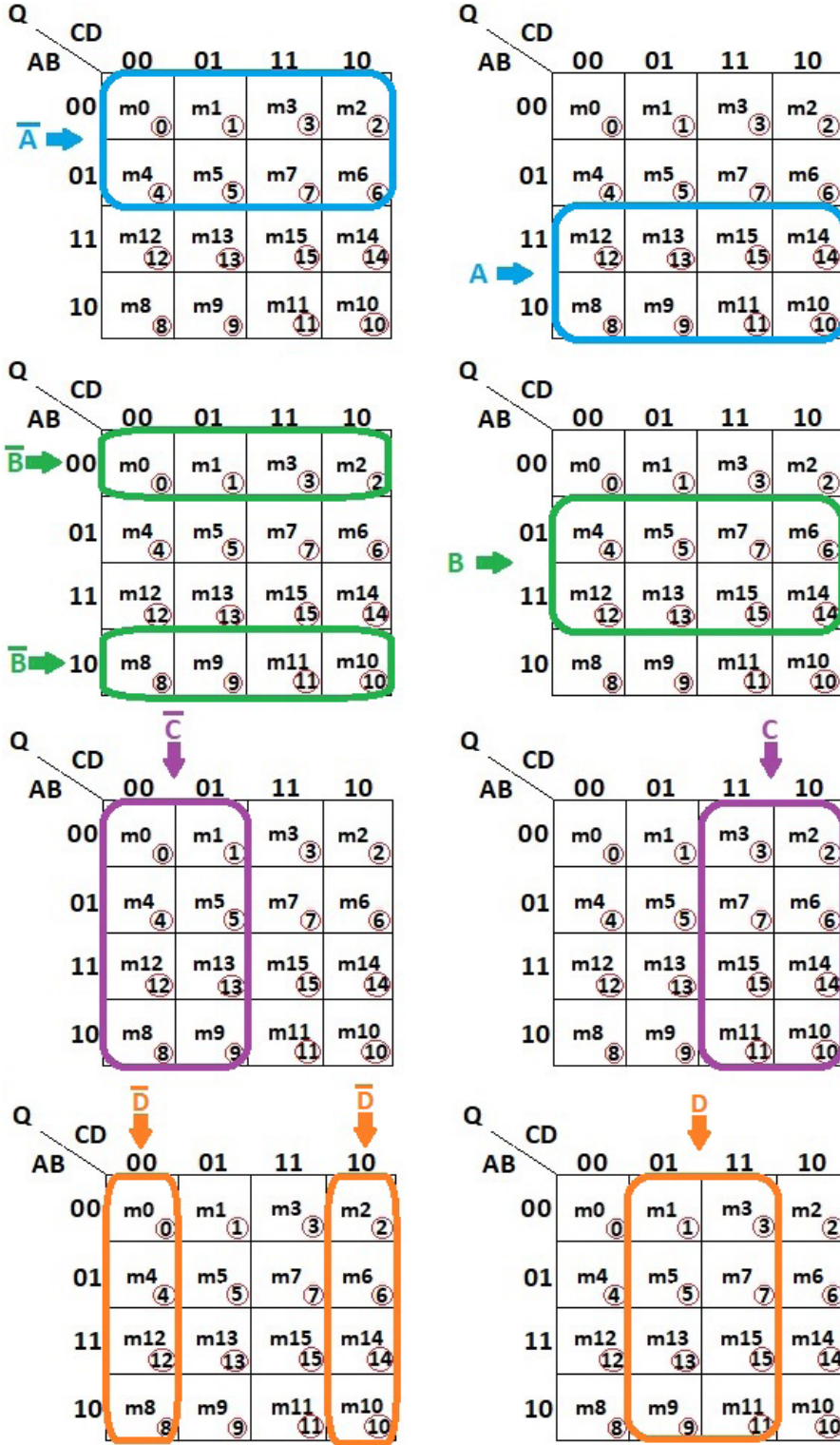
$m0 = \overline{A}\overline{B}\overline{C}\overline{D}$
 $m1 = \overline{A}\overline{B}\overline{C}D$
 $m2 = \overline{A}\overline{B}C\overline{D}$
 $m3 = \overline{A}\overline{B}CD$
 $m4 = \overline{A}B\overline{C}\overline{D}$
 $m5 = \overline{A}B\overline{C}D$
 $m6 = \overline{A}BC\overline{D}$
 $m7 = \overline{A}BCD$
 $m8 = A\overline{B}\overline{C}\overline{D}$
 $m9 = A\overline{B}\overline{C}D$
 $m10 = A\overline{B}C\overline{D}$
 $m11 = A\overline{B}CD$
 $m12 = AB\overline{C}\overline{D}$
 $m13 = AB\overline{C}D$
 $m14 = ABC\overline{D}$
 $m15 = ABCD$

c)

Görsel 1.53: a) Doğruluk tablosu Görsel 1.53: b) Karno haritası Görsel 1.53:c) Lojik ifadeler

Karno haritasında satırlar A ve B değişkenini gösterir. A değişkeninin 00 ve 01 olduğu satır, A değişkeninin tersini, 11 ve 10 olduğu satır, A değişkeninin kendisini gösterir. B değişkeninin 00 ve 10 olduğu satır, B değişkeninin tersini, 01 ve 11 olduğu sütun, B değişkeninin kendisini gösterir. Sütunlar C ve D değişkenlerini gösterir. C değişkeninin 00 ve 01 olduğu sütun, C değişkeninin tersini, 11 ve 10 olduğu sütun, C değişkeninin kendisini gösterir. D değişkeninin 00 ve 10 olduğu

sütun, D değişkeninin tersini, 01 ve 11 olduğu sütun D değişkeninin kendisini gösterir (Görsel 1.54).



Görsel 1.54: A, B, C ve D değişkenlerinin karno haritası üzerinde gösterilmesi



ÖRNEK 40

$Q = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot D + A \cdot B \cdot C \cdot \bar{D} + A \cdot B \cdot C \cdot D$ lojik ifadesini karno haritası kullanarak sadeleştiriniz.

ÇÖZÜM:

	GİRİŞLER				ÇIKIŞ
	A	B	C	D	
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

a)

Q	CD				
		00	01	11	10
AB	00	1 0	1 1	0 3	0 2
	01	1 4	1 5	0 7	0 6
	11	0 12	0 13	1 15	1 14
	10	0 8	0 9	1 11	1 10

b)

Görsel 1.55: a) Örnek 40'a ait doğruluk tablosu Görsel 1.55: b) Karno haritası

$Q = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot D + A \cdot B \cdot C \cdot \bar{D} + A \cdot B \cdot C \cdot D$ ifadesine göre Görsel 1.55.a'daki doğruluk tablosu hazırlanır.

$\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$ ifadesi karno haritasının 0.hücresinin 1 olduğunu belirtiyor. $\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D$ ifadesi karno haritasının 1.hücresinin 1 olduğunu belirtiyor. $\bar{A} \cdot B \cdot \bar{C} \cdot \bar{D}$ ifadesi karno haritasının 4.hücresinin 1 olduğunu belirtiyor. $A \cdot \bar{B} \cdot C \cdot \bar{D}$ ifadesi karno haritasının 5.hücresinin 1 olduğunu belirtiyor. $A \cdot \bar{B} \cdot C \cdot D$ ifadesi karno haritasının 10.hücresinin 1 olduğunu belirtiyor. $A \cdot \bar{B} \cdot C \cdot D$ ifadesi karno haritasının 11.hücresinin 1 olduğunu belirtiyor. $A \cdot B \cdot C \cdot \bar{D}$ ifadesi karno haritasının 14.hücresinin 1 olduğunu belirtiyor. $A \cdot B \cdot C \cdot D$ ifadesi karno haritasının 15.hücresinin 1 olduğunu belirtiyor. Diğer hücreler kullanılmadığından 0 olarak yazılır.

Karno haritasında 1'lerin oluşturacağı grupların en büyüğünü işaretlemek esas olduğundan Görsel 1.55:b'deki Q1 ve Q2 ifadesi gruplandırılır. Grupların 1, 2, 4, 8, 16 gibi ikinin katları olacağı unutulmamalıdır.

Q1 ifadesinde A, B, C ve D değişkenlerinin olup olmadığına bakılır. Q1 ifadesinde A'nın değili vardır. Q1 ifadesinde B hem 0 hem 1 değerini aldığından B değişkeni yazılmaz. Q1 ifadesinde C'nin değili vardır. Q1 ifadesinde D hem 0 hem 1 değerini aldığından D değişkeni yazılmaz. Böylece $Q1 = \bar{A} \cdot \bar{C}$ olur.

Q2 ifadesinde A, B, C ve D değişkenlerinin olup olmadığına bakılır. Q1 ifadesinde A'nın kendisi vardır. Q1 ifadesinde B hem 0 hem 1 değerini aldığından B değişkeni yazılmaz. Q1 ifadesinde C'nin kendisi vardır. Q1 ifadesinde D hem 0 hem 1 değerini aldığından D değişkeni yazılmaz. Böylece $Q1 = A \cdot C$ olur. $Q = Q1 + Q2$ olduğundan $Q = \overline{A} \cdot \overline{C} + A \cdot C$ ifadesi olarak bulunur.



UYGULAMA

Adı:	Dört Değişkenli Karno Haritası ile Bulunan Lojik İfadeye Ait Devrenin Breadboard Üzerine Kurulması	No.: 1.11
Amaç:	Temel lojik entegrelerle breadboard üzerine devre kurmak.	Süre: 20 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda $Q = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot D$ lojik ifadesi karno haritası ile sadeleştirip breadboard üzerine kurunuz.

Kullanılacak Araç Gereç: Tablo 1.12'de belirtilmiştir.

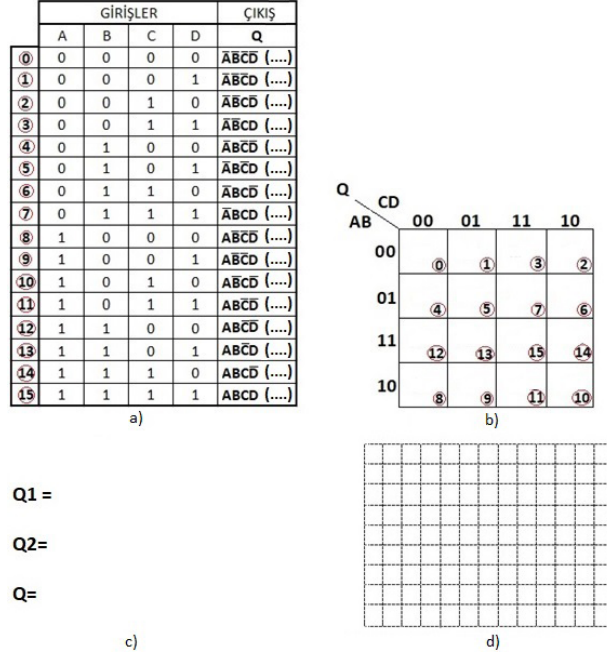
Tablo 1.12: Kullanılacak Araç Gereç Listesi

Adı	Özelliği	Miktarı
Entegre	7408, 7432, 7404	3 Adet
Direnç	470Ω	5 Adet
DIP anahtar	Dörtlü	1 adet
LED	Kırmızı	1 adet
Breadboard		1 adet
Güç kaynağı	5 volt	1 adet
Kablo	Bağlantı kablosu	20 adet

Uygulama Adımları:

1. Adım: Tablo 1.12'de belirtilen malzemeleri hazırlayınız.

2. Adım: $Q = \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot B \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot D$ lojik ifadesine göre Görsel 1.56.a'daki doğruluk tablosunun Q çıkış değerlerini hesaplayınız.



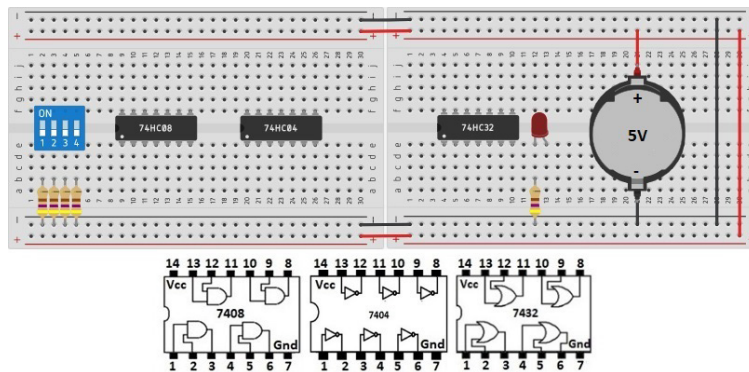
Görsel 1.56: a) Uygulama 1.11'e ait doğruluk tablosu Görsel 1.56: b) Karno haritası Görsel 1.56: c) Lojik ifade Görsel 1.56: d) Lojik diyagram

3. Adım: Doğruluk tablosunda 1'leri dikkati alarak Görsel 1.56:b'deki karno haritasını doldurunuz. Karno haritasına 1'leri doğru yerleştirdiyseniz Q1 ve Q2 olmak üzere iki grup bulacaksınız.

4. Adım: Karno haritasında grupları sadeleştirerek Q1, Q2 ve $Q = Q1 + Q2$ olarak bulduğunuz lojik ifadeleri Görsel 1.56:c'de verilen bölüme yazınız.

5. Adım: Bulduğunuz lojik ifadenin lojik şemasını Görsel 1.56:d'de verilen kısma çiziniz.

6. Adım: Çalışma ortamında dikkati dağıtacak malzemeleri bulundurmamaya dikkat ediniz. İş güvenliği tedbirlerini alarak uygulama Görsel 1.57'de görüldüğü gibi devre elemanlarını breadboard üzerine yerleştiriniz.



Görsel 1.57: Breadboard üzerine kurulacak uygulama devresi

7. Adım: Bağlantı kablolarını lojik şemaya uygun olarak breadboard üzerine yerleştiriniz.

8. Adım: Öğretmeninizden onay aldıktan sonra güç kaynağını 5 volta ayarlayınız ve breadboard üzerine Görsel 1.57’da görüldüğü gibi bağlayınız.

9. Adım: A, B, C ve D girişlerini Görsel 1.56:a’da gösterildiği gibi devreye uygulayarak, lojik devrenin doğruluk tablosunun doğru hazırlanıp hazırlanmadığını kontrol ediniz.

10. Adım: Elde ettiğiniz sonuçları öğretmeninize kontrol ettiriniz.

11. Adım: Çalışmanız bitince güç kaynağını kapatınız.

12. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmanız diğer sayfadaki kontrol listesi kullanılarak değerlendirilecektir. Çalışmanızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Doğruluk tablosu doğru hazırlandı.		
3. Karno haritası doğru yerleştirildi.		
4. Sadeleştirme doğru yapıldı.		
5. Breadboard üzerine lojik devre doğru kuruldu.		
6. Breadboard üzerine kurulan devre doğruluk tablosundaki Q çıkış değerlerine uygun çalıştırıldı.		
7. Temizliğe dikkat edildi.		



SIRA SİZDE

$Q = \overline{A} \cdot \overline{B} \cdot C \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot C \cdot D$ lojik ifadesini karno haritası kullanarak sadeleştiriniz. Lojik devresini breadboard üzerine kurunuz.



ÖLÇME VE DEĞERLENDİRME 1

A) Aşağıdaki cümlelerde parantezlerin içine yargılar doğru ise (D), yanlış ise (Y) yazınız.

1. () 0-7 arası sayılarından oluşan sayı sistemine sekizlik sayı sistemi denir.
2. () İkili toplam işleminde elde varsa sağdaki bitlere aktarılır.
3. () Girişine uygulanan lojik bilgilerin çarpımını alan lojik kapıya VE kapısı denir.
4. () A lojik ifadesinin değili \bar{A} olarak gösterilir.
5. () VEDEĞİL kapısı VE ve DEĞİL kapısının lojik toplamından elde edilir.

B) Aşağıdaki cümlelerde boş bırakılan yerlere doğru sözcükleri yazınız.

6. 0-15 arası sayılarından oluşan sayı sistemine sayı sistemi denir.
7. İkili sayı sisteminde en değersiz bit olarak adlandırılır.
8. Girişine uygulanan lojik bilgilerin toplamını alan lojik kapıya kapısı denir.
9. $Q = A(A + 1)$ işleminin sonucu olarak hesaplanır.
10. TTL lojik kapılar gerilim altında çalışırlar.

C) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

11. Onaltılık (Hexadecimal) sayı sistemine ait sayı aşağıdakilerden hangisidir?

- A) $(01H)_{16}$ B) $(1CL)_{16}$ C) $(77X)_{16}$ D) $(9AJ)_{16}$ E) $(A1F)_{16}$

12. $(11)_2 + (10)_2$ ikilik sayıların toplam sonucu aşağıdakilerden hangisidir?

- A) $(100)_2$ B) $(101)_2$ C) $(110)_2$ D) $(111)_2$ E) $(1010)_2$

13. $(0110110011)_2 = (?)_{16}$ ikilik sayının onaltılık karşılığı aşağıdakilerden hangisidir?

- A) $(1B3)_{16}$ B) $(2B3)_{16}$ C) $(2C3)_{16}$ D) $(3B3)_{16}$ E) $(6C3)_{16}$

14. DEĞİL (NOT) kapısının görevi aşağıdakilerden hangisidir?

- A) Girişine uygulanan lojik veriyi aynen çıkışına aktarır.
- B) Girişine uygulanan lojik veriyi sıfır ile çarpıp çıkışına aktarır.
- C) Girişine uygulanan lojik verinin tersini çıkışına aktarır.
- D) Girişine uygulanan lojik veriyi bir ile çarpıp çıkışına aktarır.
- E) Girişine uygulanan lojik verinin değilinin değilini alarak çıkışına aktarır.

2. ÖĞRENME BİRİMİ



MİKRODENETLEYİCİ PROGRAMLAMA



KONULAR

- 2.1. MİKRODENETLEYİCİ VE PROGRAMI
- 2.2. MİKRODENETLEYİCİYLE GİRİŞ-ÇIKIŞ KONTROLÜ
- 2.3. MİKRODENETLEYİCİYE PROGRAMI YÜKLEYEREK TEST ETME

NELER ÖĞRENECEKSİNİZ?

- Mikro işlemci ve mikrodnetleyici arasındaki farklar
- Mikrodnetleyicinin temel bileşenleri
- *Mikrodnetleyici programlama dili
- Mikrodnetleyicinin giriş-çıkış ayarları
- Mikrodnetleyici programlama yazılımının kurulumu
- Mikrodnetleyici programlama dili ile temel program yazılımı
- Mikrodnetleyici programlama yazılımını kullanarak denetleyicinin giriş-çıkış portlarının kontrolü
- Mikrodnetleyici programlama yazılımında yazılan programı derleme
- Derlenen yazılımı mikrodnetleyiciye yükleme

ANAHTAR KELİMELE

ALU, CISC, CPU, DDRx, EEPROM, FLASH, Havard, I/O, kaydediciler, Mikro işlemci, mikrodnetleyici, osilatör devreleri, PINx, Port, PORTx, program belleği, reset devresi, RISC, SRAM, veri belleği, Von Neumann.

HAZIRLIK ÇALIŞMALARI

1. Mikro işlemci ve mikrodnetleyici arasındaki farklar nelerdir?
2. Çevrenizde mikrodnetleyicilerle çalışan cihazlar nelerdir?

2.1. MİKRODENETLEYİCİ VE PROGRAMI

Önceden belirlenmiş özel amaçları yerine getirmek için tasarlanmış ve belirlenen görevlerin dışına çıkamayan bilgisayar sistemlerine, **gömülü bilgisayar sistemleri (Embedded Computing System)** adı verilir. Gömülü sistemler, belirli bir amaca yönelik olduklarından tasarım mühendisleri tarafından ürünün boyutu ve kullanılan donanımlar ihtiyaçlara uygun tasarlanır ve maliyet düşürülebilir. Gömülü sistemlerde de bilgisayar sistemlerinde olduğu gibi CPU, RAM, ROM, I/O vb. birimler kullanılır fakat bu sistemlerde cihaz, belirli görevleri yerine getirdiği için çok büyük boyutlarda hız gerektiren birimlere ihtiyaç duyulmaz.

Günümüzde, gömülü bilgisayar sistemine sahip; buzdolabı, çamaşır makinesi, TV, dijital fotoğraf makinesi, fotokopi makinesi, otomobillerde otomatik vites sistemleri ve yol bilgisayarı, akıllı ev sistemleri, dronlar (drone) gibi birçok cihaz bulunmaktadır (Görsel 2.1).



Görsel 2.1: Gömülü sistemler

Gömülü sistemleri daha iyi kavrayabilmek için mikro işlemci ve mikrodenetleyici kavramlarını anlayabilmek gerekir.

2.1.1 Mikro İşlemci

Bilgisayar sistemlerinin beyni olarak adlandırılan mikro işlemciler, programcı tarafından yazılan programların yapmak istediği tüm işlemleri yerine getirir. Aritmetik ve mantıksal işlemleri yerine getirebilen ve bu işlemlerin sonucuna göre işlem akışını yönlendirebilen tümleşik bir devredir.

Merkezî işlem birimi (MİB), işlemci ya da CPU (Central Processing Unit) olarak da adlandırılır (Görsel 2.2).



Görsel 2.2: Mikro işlemci

Mikro işlemciler; çok hızlı bir hesap makinesi gibi düşünülebilir, sadece 0 ve 1 (binary sayı sistemi) değerleri üzerinden işlem yapabilir. Güncel hızları GHz (Gigahertz) seviyelerindedir.

Bir mikro işlemci tek başına kendisinden istenen görevleri yerine getiremez. Bazı yardımcı donanımlara ihtiyaç duyar. Bunlar;

- Giriş (Input),
- Çıkış (Output),
- Hafıza (Memory) birimleridir.

Bu birimler mikro işlemcinin dışında anakart denilen elektronik kart üzerinde yer almaktadır. Ayrıca anakart üzerinde yer alan iletişim yolları sayesinde bu birimler birbiri ile haberleşmektedir.

İşlemciye dış dünyadan veri transferi sağlayan birim, giriş (Input) birimidir. Klavye, fare, mikrofon, tarayıcı gibi cihazlar giriş birimlerine örnektir. Veri, işlemciye insan, başka bir bilgisayar, elektronik bir sistem, sensörler vb. birçok kaynaktan gelebilir. İşlemci üzerinde veriler işlendikten sonra çıkan sonuçlar direkt çıkış birimlerine (monitör, yazıcı, hoparlör vb.) aktarılabilir. Bilgisayar sistemlerinde hafıza birimlerini genel olarak kalıcı ve geçici hafıza olarak ikiye ayırabiliriz. Kalıcı hafızalar sistemin enerjisi kesildiğinde üzerinde yer alan verileri koruyan birimlerdir. Sabit disk, CD, DVD, FLASH bellek ve ROM kalıcı hafıza birimlerine örnek verilebilir. Geçici hafızalar ise sistemin enerjisi kesildiğinde üzerinde yer alan verilerin silindiği hafıza birimleridir. RAM bellek, cache (önbellek), sanal bellek bilgisayar sistemlerinde kullanılan geçici hafıza birimlerindendir.

2.1.1.1. Mikro İşlemci Temel Birimleri

Mikro işlemci, bilgisayar sistemlerinin beyni olarak komutları işler, verileri idare eder ve diğer birimlerle olan etkileşimleri kontrol eder. Bir mikro işlemcinin yapısında Aritmetik ve Mantık Birimi (ALU), Kontrol Birimi ve Kaydediciler bulunmaktadır.

Aritmetik ve Mantık Birimi: İşlemciye gelen veriler üzerinde aritmetik ve mantıksal işlemler gerçekleştiren birimdir. Bu birim toplama, çıkarma, karşılaştırma, kaydırma ve döndürme işlemlerini yapar.

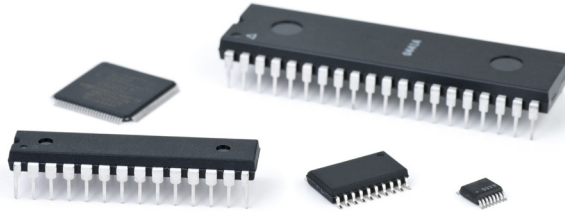
Kontrol Birimi: Bu birimde işlemci içerisindeki işleri organize etmek için kontrol sinyalleri üretilir. İşlemciye gelen komutlar çözülerek komutun ne istediği anlaşılır ve yorumlanır. Kaydedicilere hangi veriler üzerinde işlem yapılacağı söylenir. İşlem birimine veriler üzerinde hangi işlemlerin yapılacağı iletilir.

Kaydediciler: Aritmetik ve mantık biriminin üzerinde işlem yapacağı verilerin saklandığı birimdir. Ayrıca aritmetik ve mantık işlemleri sonucunda ortaya çıkan sonuçlar da bu birimlerde saklanır.

2.1.2. Mikrodenetleyici

Mikrodenetleyici, yapısında CPU (mikro işlemci), bellek (EPROM, FLASH), programlanabilir giriş-çıkış, analog-dijital dönüştürücü, sinyal üretici, sayıcı, iletişim arayüzleri gibi birimleri barındıran tümleşik tek bir devre olarak üretilen bir mikro bilgisayardır. Bir bilgisayar sistemini oluşturan temel birimleri üzerinde barındıran entegrelerdir. Günümüzde otomobil, telefon, yazıcı ve fotokopi makinesi, televizyon, kamera, drone, beyaz eşya vb. birçok cihazda kontrol sistem aygıtı olarak mikrodenetleyiciler kullanılmaktadır (Görsel 2.3).

Mikro işlemci ile bir dijital sistem oluştururken hafıza birimleri başta olmak üzere, giriş çıkış birimleri, iletişim yollarını içeren bir kart sistemi ilave ederek oluşturulabilir. Fakat mikrodenetleyici bir bilgisayar sistemin de bulunması gereken temel birimlerin tamamını kendi bünyesinde barındıran bir mikro bilgisayar sistemidir.



Görsel 2.3: Mikrodenetleyiciler

Tablo 2.1’de mikro işlemci ve mikrodenetleyicilerin karşılaştırması verilmiştir.

Tablo 2.1: Mikro işlemci ve Mikrodenetleyici Karşılaştırması

Özellik	Mikro işlemci (CPU)	Mikrodenetleyici
Program-veri belleği	Aynı bellek bloku	Farklı bellek bloku
Hız	Düşük	Yüksek
Saat çevrimi	1 komut-birden fazla çevrim	1 komut-1 çevrim
Komut sayısı	100’den fazla	50’den az
Güç tüketimi	Fazla	Az
Haricî donanım	Gerektirir	Gerektirmez
Mimari	CISC-Von Neuman	RISC-Harvard
Fiyat	Pahalı	Ucuz
Kullanım alanları	Kişisel bilgisayarlar	Gömülü sistemler

Mikro işlemci ile kontrol edilecek bir sistem kurmak için minimum **CPU, RAM, I/O** ve **kalıcı depolama birimlerine** ihtiyaç vardır. Bu birimler arasında iletişimi sağlayacak **veri yolu, adres yolu** ve **kontrol yolundan** oluşan iletişim yollarına ihtiyaç vardır. Bu yolları ve diğer birimleri bir arada barındıracak elektronik baskı devreye ihtiyaç vardır. Bu elektronik baskı devreye **mainboard** yani **anakart** adı verilmektedir.

Mikrodenetleyicili sistemlerde ise **CPU, RAM, I/O, ROM** gibi yapılar ve bu birimlerin iletişimini sağlayan iletişim yolları, tek bir entegre içerisinde barındırılır. Tek bir chip (entegre) ile kurulacak gömülü sistemlerin maliyeti daha düşüktür. Ayrıca az sayıda komutla programlama yapabilmek yazılımcı için önemli bir avantajdır. Bahsettiğimiz bu sebeplerden günümüzde bilgisayar kontrolü gerektiren sistemlerde mikrodenetleyiciler daha çok tercih edilmektedir.

2.1.2.1. Mikrodenetleyicinin Temel Bileşenleri

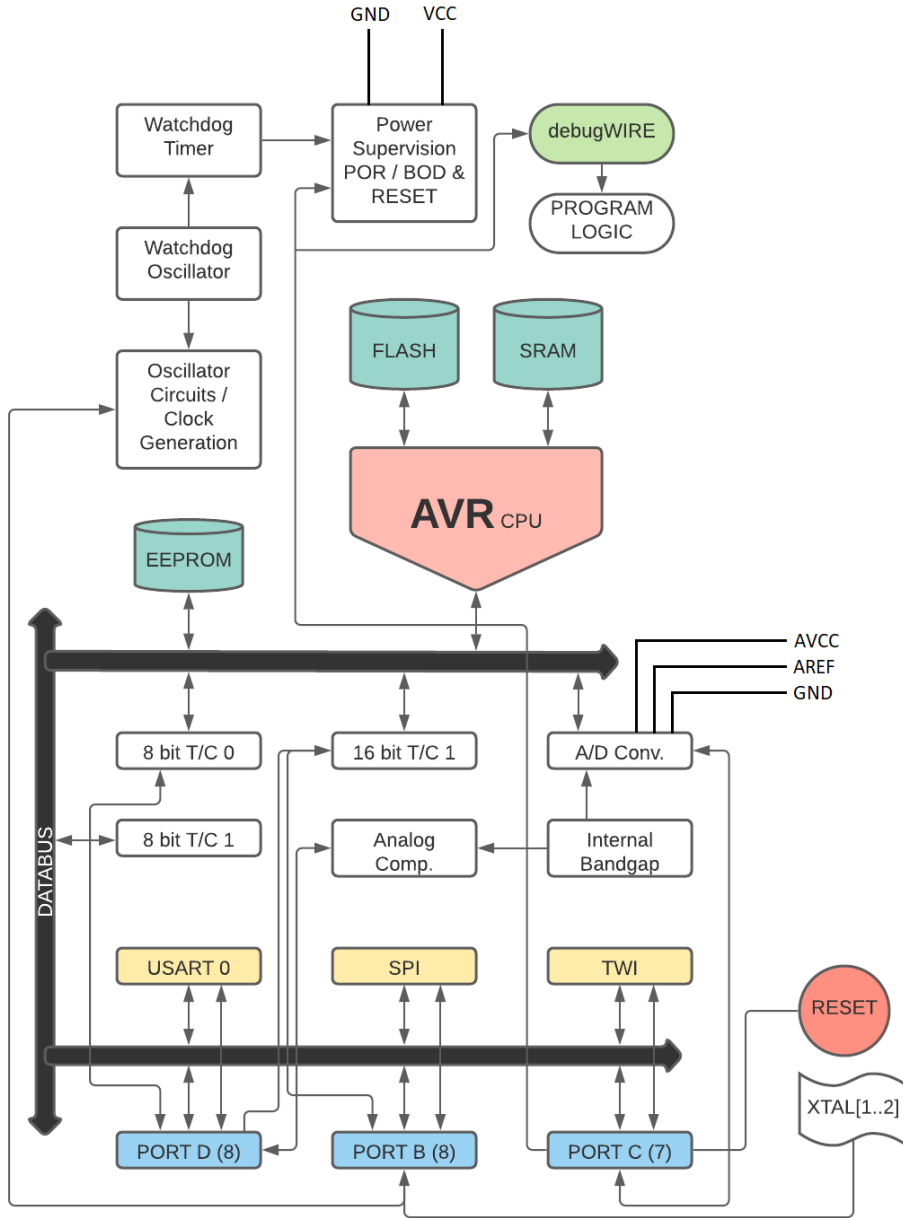
Bu ve takip eden öğrenme birimlerinde ATMEL firmasının Atmega328P mikrodenetleyicisinin temel donanımı hakkında bilgi verilecektir. Bu bilgiler diğer tüm mikrodenetleyiciler için genel bilgileri içermektedir.

Bir mikrodenetleyicinin üreticisi tarafından yayınlanan ve denetleyici hakkındaki tüm teknik bilgileri içeren bir kitapçığı bulunur. Bu kitapçıklara **datasheet (veri sayfası)** adı verilir. Bir mikrodenetleyiciyi programlamak için bu veri sayfalarındaki teknik bilgilere ihtiyaç duyulur. Mikrodenetleyicilerin veri sayfalarına üretici firmaların web sayfalarından / adreslerinden ulaşılabilir.

Bir mikrodenetleyicinin yapısında aşağıda verilen temel bileşenler bulunmaktadır.

- CPU Çekirdeği
- SRAM
- FLASH

- EEPROM
- Giriş / Çıkış Portları
- Timer (Zamanlayıcı)
- İletişim Arayüzleri (UART-USART-I2C-SPI)



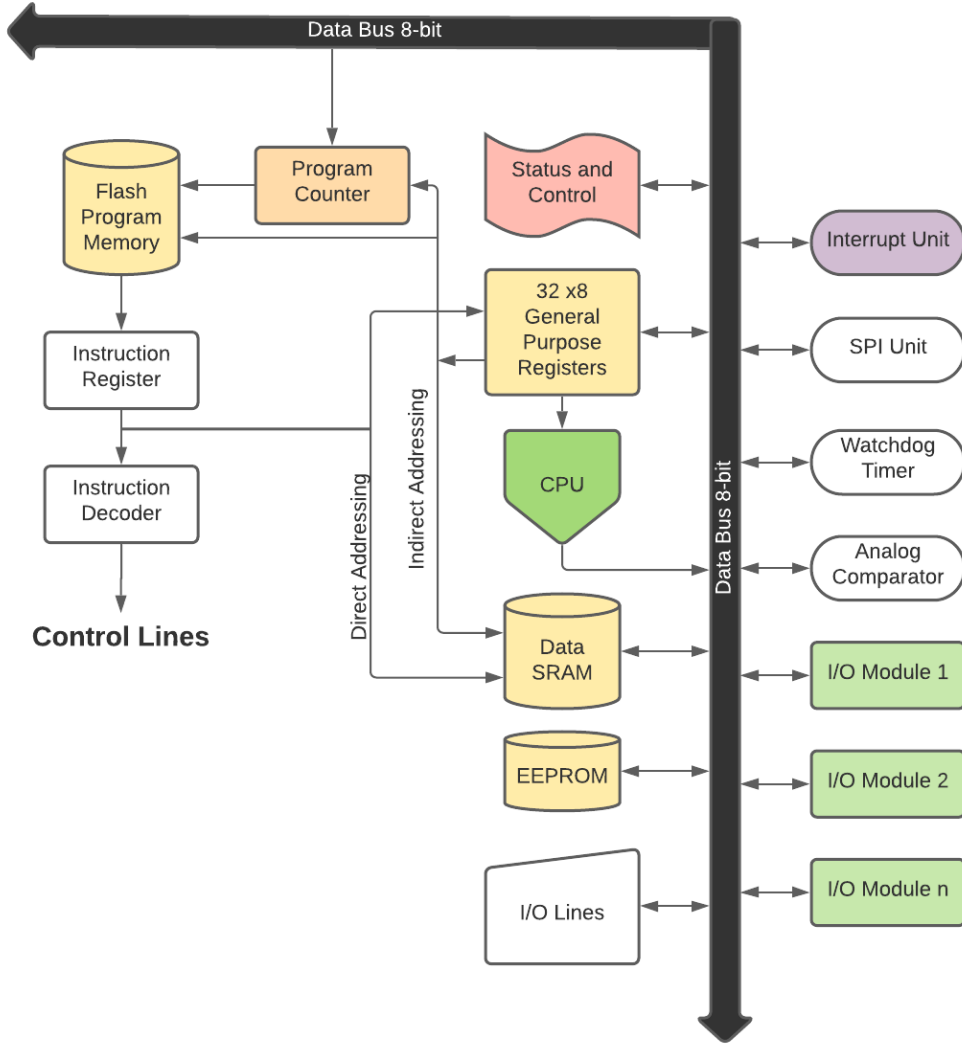
Görsel 2.4: Atmega328P blok diyagramı

Görsel 2.4'te Atmega328P mikrodenetleyicinin blok diyagramı verilmiştir. Blok diyagramı, bir mikrodenetleyicinin iç yapısında bulunan birimleri ve bu birimlerin birbiriyle olan bağlantılarını gösterir. Programcı için denetleyicinin blok diyagramını okuyabilmek oldukça önemlidir.

Görsel 2.4'teki blok diyagramından hareketle mikrodenetleyicilerin temel bileşenleri aşağıdaki başlıklardaki gibidir.

CPU Çekirdeği

Bir mikrodenetleyici işlemcisinin (CPU) temel görevi, tüm denetleyici birimlerini organize etmektir. Bu nedenle CPU belleklere erişebilmeli, gerekli hesaplamaları ve mantıksal işlemleri yapabilmeli, denetleyicinin çevre birimlerini kontrol edebilmeli ayrıca gelen kesmeleri yürütebilmelidir (Görsel 2.5).



Görsel 2.5: Atmega328P mikrodenetleyicisinin CPU blok diyagramı

CPU program belleğinde bulunan program komutlarını ardışık bir sıra ile doğrusal olarak işler. İşlemci içerisinde bir komut yürütülürken program belleğindeki sıradaki komut alınır. İşlemci sıradaki komutun adresini Program Counter denilen kaydedici üzerinden alır.

İşlemci her saat darbesinde sıradaki komut okunduktan sonra Program Counter'ın değeri bir arttırılır. Kısaca PC (Program Counter) CPU içerisinde icra edilecek bir sonraki komutun adres verisini tutar.

ALU [Aritmetik ve Lojik Birimi (Arithmetic Logic Unit)]: İşlemci temel birimlerinden biri olan ALU, aritmetik, mantıksal ve bit işlemleri olmak üzere üç ana fonksiyonu yerine getirir.

Aritmetik işlemler denildiğinde akla toplama, çıkarma, çarpma ve bölme; mantıksal işlemler denildiğinde AND, OR, EXOR ve NOT; bit işlemleri denildiğinde ise kaydırma, tümleyen alma vb. işlemler gelir. Tüm bu işlemler, CPU içerisinde ALU'da mantık kapıları ile oluşturulmuş devreler vasıtasıyla gerçekleştirilir. Bu devreler dâhilî veri yoluyla birbirine bağlıdır. Aynı şekilde özel bir veri yoluyla da kaydedicilere bağlıdır.

Atmega328P mikrodenetleyici işlemcisi (CPU) içerisinde 32 adet genel amaçlı kaydedici (General Purpose Registers) bulunmaktadır. Bu kaydediciler 8 bitlik (1 bayt) veri saklayabilir. Bu kaydediciler SRAM yapıdaki veri belleğinin ilk bölümünde yer almaktadır. ALU bu kaydedicilere doğrudan bağlantılı olarak çalışır. Bu durum yüksek performans sağlar.

ALU veriyi kaydedicilerden alır, işletir ve sonucu kaydedicilere kaydeder. Kontrol birimi ALU'nun veri üzerinde yapacağı işlemi seçer. ALU içerisinde gerçekleşen bütün bu işlemler kontrol biriminin ürettiği kontrol sinyalleri sayesinde gerçekleştirilir.

Registers (Kaydediciler)

Atmega328P mikrodenetleyicisi 8 bitlik bir mikrodenetleyicidir. Burada kullandığımız 8 bit ifadesi, mikrodenetleyicinin kendi iç yapısında bulunan dâhilî kaydedici birimleriyle yaptığı iletişimde kullandığı veri yolunun bit sayısını ifade eder. Bu bit sayısı, kelime boyu (Word Length) olarak adlandırılır. Denetleyici, hafızasında verileri sekizer bitlik alanlarda saklar.

Atmega328P denetleyicisinin veri hafızasında (SRAM), genel amaçlı ve özel amaçlı olmak üzere sekizer bitlik, denetleyici işlemcisinin direkt bağlı olduğu çeşitli alanlar bulunur. Bu alanlara **kaydedici (register)** adı verilir.

General Purpose Registers (Genel Amaçlı Kaydediciler): Genel amaçlı kaydediciler CPU içerisinde verilerin geçici olarak saklandığı alanlardır. ALU birimi bu kaydedicilere doğrudan bağlantılıdır. Bu kaydedicilere veri bellek alanından ya da çevre birimlerden gelebilir. 8 bit şeklinde organize edilebilecekleri gibi $8+8=16$ bitlik alan olarak da organize edilebilir.

Genel amaçlı kaydediciler sayesinde CPU'nun sürekli Veri belleğine (SRAM) gitmesinin önü kesilmiş olur. Aynı şekilde yapılan işlemlerin sonuçları da öncelikle genel amaçlı kaydediciler üzerine kaydedilir. Böylelikle CPU'nun performansı maksimuma çıkarılmış olur.

Atmega328P mikrodenetleyicisinde 32 adet 8 bitlik genel amaçlı kaydedici bulunmaktadır. Bu kaydediciler veri belleğinde 0x0000-0x001F adreslerinde yer alır. Bir CPU içerisinde ne kadar çok genel amaçlı kaydedici varsa işlemleri o kadar hızlı yürütmektedir.

Özel Amaçlı Kaydediciler: Özel amaçlı kaydediciler denetleyicinin çeşitli durum ve ayarlarının tutulduğu özel kaydedicilerdir. Örneğin bir aritmetik ve mantıksal işlem sonucunda ortaya çıkan

durumların kaydedildiği Status Register (SREG), işlemcinin hangi komutu işleteceğinin adresini tutan Program Counter (PC), portların yön bilgilerini tutan DDRx kaydedicileri gibi. Atmega328P denetleyicisinde bu kaydediciler veri belleğinde 0x20-0xFF adres aralığında yer almaktadır. Aşağıda bazı özel amaçlı kaydedicilerin açıklaması yapılmıştır. Konular ilerledikçe konuların ilişkili olduğu kaydediciler konu içerisinde anlatılacaktır.

SREG [Status Register (Durum Kaydedicisi)]: Durum kaydedicisi son yapılan aritmetik işlemin sonucunda oluşan bilgilerin kaydedildiği bellek alanıdır. Yapılan tüm ALU işlemleri sonucunda içerisindeki veriler güncellenir. 8 bitlik bir veri alanına sahiptir. Her bir bit alanının farklı bir anlamı vardır (Görsel 2.6).

Bit	7	6	5	4	3	2	1	0	Adres
Bayraklar	I	T	H	S	V	N	Z	C	0x3F (0x5F)
Okuma/Yazma	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Varsayılan Değer	0	0	0	0	0	0	0	0	
SREG – Status Register – Durum Kaydedicisi									

Görsel 2.6: SREG [Status Register (Durum Kaydedicisi)]

7. Bit [I (Interrupt Enable)]: Kesme yetkilendirme bayrağı olarak adlandırılır. Mikrodenetleyiciye bağlı harici cihazlardan ya da mikrodenetleyici birimlerden gelen kesme taleplerine izin verilip verilmeyeceğini belirler. Interrupt bitinin 0 (sıfır) olması gelen kesme isteklerine cevap verilmemesini sağlar. Interrupt biti 1 (bir) ise gelen kesme isteklerine cevap verilir ve ilgili kesme işletilir.

6. Bit [T (Bit Copy Storage)]: Bit kopyalama bayrağı olarak adlandırılır. Bu bayrak, BLD (Bit Load) ve BST (Bit Store) komutları için anlamlıdır. Bitin kaynak mı hedef mi olduğunu belirtir.

5. Bit [H (Half Carry Flag)]: Yarım taşma bayrağı, yardımcı bayrak veya ondalık ayar bayrağı olarak adlandırılır. Bir matematiksel işlemin yürütülmesi sonucu akümülatör kaydının en az önemli dört bitinden bir taşma ya da borç oluşması durumunda değeri 1 (bir) olur. Daha çok BCD aritmetik işlemlerinde kullanılır.

4. Bit [S (Sign Bit)]: İşaret bayrağı olarak adlandırılır. İşaretleli sayılarla yapılan işlemlerde bu bayrak anlam ifade eder. Aritmetik, mantıksal, kaydırma ve yönlendirme işlemlerinin sonucu **negatif** çıkarsa bu bayrağın değeri 1 (bir); **pozitif** çıkarsa değeri 0 (sıfır) olur.

3. Bit [V (Two's Complement Overflow Flag)]: İşaretleli tam sayılarla toplama ve çıkarma işlemi yapıldığında taşma ya da borç oluştuğunda bu bayrak set edilmektedir yani 1 olmaktadır.

2. Bit [N (Negative Flag)]: Negatif bayrağı olarak adlandırılır. Bir aritmetik ya da mantıksal işlemin sonucu negatif çıkarsa N bayrağının değeri 1, pozitif çıkarsa 0 değerini tutar.

1. Bit [Z (Zero Flag)]: Sıfır bayrağı olarak adlandırılır. Bir aritmetik ya da mantıksal işlemin sonucu 0 (sıfır) çıkarsa Z bayrağının değeri 1, aksi hâlde 0 olacaktır.

0. Bit [C (Carry Flag)]: “Elde Bayrağı” ya da “Taşma Bayrağı” olarak adlandırılır. Bir toplama işleminde elde, çıkarma işleminde ise borç oluşursa C bayrağının değeri 1 aksi hâlde 0 olur. Ayrıca çarpma işleminde sonuç gösterici görevini üstlenir. Kaydırma ve yönlendirme işlemlerinde ise MSB ve LSB bitlerinden düşen bit değerlerini tutar.

Stack Pointer (Yığın İşaretçisi)

Yığın, geçici verileri depolamak, program içerisinde yer alan yerel değişkenlerin içeriklerini depolamak için kullanılan bellek alanıdır. Yığın bellek alanı, SRAM (Veri Belleği) içerisinde yer alır. Bu işaretçi, 16 bitlik bir kaydedicidir, yığın alanına veri yazılacaksa yığın alanındaki boş alanlarının adreslerini; yığın alanından veri okunacaksa okunacak verinin adres verisini tutar. Atmega328P denetleyicisinde Stack Pointer kaydedicisi SRAM içerisinde 0x3E-0x3D adreslerinde yer alır (Görsel 2.7).

Bit	15	14	13	12	11	10	9	8	Adres
SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	0x3E
SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	0x3D
Bit	7	6	5	4	3	2	1	0	
Okuma/Yazma	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Varsayılan Değer									

Stack Pointer Registers – Yığın İşaretçi Kaydedicileri

Görsel 2.7: Stack Pointer Kaydedicileri

Mikrodenetleyici Bellek Organizasyonu

Bir mikrodenetleyicinin yapısında “SRAM Bellek” (Veri Belleği), “FLASH Bellek” (Program Belleği) ve “EEPROM Bellek” olmak üzere üç çeşit bellek bulunur. Mikrodenetleyicilerde bulunan bellekler bilgisayar sistemlerindeki gibi GB, TB boyutlarında hafızalar değildir. Bu hafızalar, mikrodenetleyicilerde birkaç KB boyutundadır. Atmel Atmega328P mikrodenetleyicisi iki adet ana bellek alanına sahiptir. Bu bellekler veri belleği (SRAM) ve program belleğidir (FLASH). Ayrıca haricî verilerin depolanması için EEPROM belleğe sahiptir. Bu üç bellek alanı da doğrusal ve düzenli bir yapıya sahiptir.

Atmega328P mikrodenetleyicisinde performansı en üst seviyeye çıkarmak için ayrı bellek blokları kullanılmıştır. Bu yapı sayesinde program ve verilere aynı anda erişilebilmektedir. Bu yapıya Harvard mimarisi adı verilir.

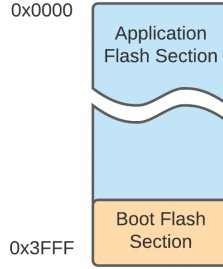
Atmega328P mikrodenetleyicisinin bellek organizasyonunu aşağıdaki başlıklardaki gibidir.

FLASH Bellek [Program Belleği (32 KB)]: Program komutlarının tutulduğu hafıza alanıdır. Bu alanda yazılan programın makine diline derlenmiş biçimi bulunur. Bu bellek türü FLASH bellek yapısındadır. Enerji kesildiğinde içindeki veriler silinmez.

FLASH bellek mikrodenetleyicinin yapmasını istediğimiz işlemlerin kodlandığı programın kodlarının saklandığı programlanabilir bellektir. Bu bellek alanına sadece program yüklenebilmektedir. Denetleyicinin çalışması esnasında bu bellek alanına erişim sağlanamamaktadır. Atmega328P mikrodenetleyicisinin üzerinde dâhilî 32 KB’lık FLASH bellek

bulunmaktadır. 16 bit genişliğine sahiptir ve 10 000 (on bin) kez yazılıp silinebilme özelliğine sahiptir.

Atmega328P mikrodenetleyicisinde program belleği yazılım güvenliği açısından iki bölüme ayrılmıştır. Bu bölümler “Uygulama Programı Bölümü” (Application Flash Section) ve “Önyükleyici (Boot Flash Section) Bölümü”dür (Görsel 2.8).



Görsel 2.8: Atmega328P Program belleği haritası

Program Counter [Program Sayıcı (PC)]: Mikrodenetleyicilerde “Program Belleği”ne (FLASH) erişim, “Program Counter” (PC) yani program sayıcıyla sağlanır. PC’nin uzunluğu mikrodenetleyici marka ve modeline göre değişiklik gösterebilir. Mikrodenetleyicide FLASH belleği adreslemeyi PC yapar.

Atmega328P mikrodenetleyicisi 14 bit genişliğinde Program Counter’a sahiptir. 14 bitlik bir PC ile $2^{14} = 16\,384$ farklı adres alanı adresleyebilir. Bu durumda 16 kilobayt [$16\,384 \times 8$ (Bir adres alanı 8 bitlik veri tutmaktadır.)] bellek alanına erişilebileceğini belirtmektedir.

Mikrodenetleyicinin bellek hücrelerini okuyabilmesi için saat sinyaline ihtiyaç vardır. Saat sinyali mikrodenetleyicilere osilatör devreleriyle sağlanır. PIC Mikrodenetleyicilerde her dört saat sinyalinde PC içerisindeki adres verisi bir artmaktadır. Atmega mikrodenetleyicilerde ise her saat sinyalinde PC içeriği bir artmaktadır. Yani PIC mikrodenetleyicilerde bir komut dört saat sinyalinde işlenirken Atmega mikrodenetleyicilerde bir saat sinyalinde bir komut işlenmektedir. Program kodu sonlandığında ya da mikrodenetleyici reset atıldığında PC içeriği program belleğinin ilk hücresi olan 0x0000 adresini gösterir.

SRAM Bellek [Veri Belleği (2 KB)]: Statik RAM yapıdaki bu bellek alanı, geçici verilerin saklandığı denetleyicinin enerjisi kesildiğinde içerisindeki verilerin silindiği alandır (Görsel 2.9).

Data Memory	
Adres	Bellek
0x0000-0x001F	32 Registers
0x0020-0x005F	64 I/O Registers
0x0060-0x00FF	160 Ext. I/O Registers
0x0100-0x08FF	Internal SRAM (1048 x 8)

Görsel 2.9: SRAM bellek haritası

Genel amaçlı ve özel amaçlı kaydediciler bu bellek alanında 0x0000 ve 0x00FF adreslerinde yer alır. **Genel amaçlı kaydediciler**, programcı tarafından yazılım geliştirilirken kullanılan kaydedicilerdir. Örneğin, bir yazılım içerisinde oluşturulan değişkenler derleyici tarafından bir bellek alanında temsil edilir. Bu bellek alanlarına genel amaçlı kaydedici adı verilir.

Özel amaçlı kaydediciler ise denetleyicinin donanımını kontrol eden kaydedicilerdir. Denetleyicinin CPU ve çevre birimlerinin doğru bir şekilde çalışması için gerekli olan ayarların saklandığı özel bellek alanları, özel amaçlı kaydedicilerdir. Örneğin, D portuna bağlı pinlerin hangilerinin giriş, hangilerinin çıkış olduğunu tutan DDRD kaydedicisi gibi. Veri belleği içerisinde 64 adet Giriş / Çıkış kaydedicisi yer almaktadır.

EEPROM Bellek (1 KB): Atmel Atmega328P mikrodenetleyicisinin dâhilî 1 KByte / kilobayt / KB boyutunda EEPROM belleği bulunmaktadır. Bu bellek alanındaki veriler, enerji kesildiğinde silinmez. 100.000 kez yazma / silme özelliğine sahiptir. EEPROM belleklerde veri on yıl kadar silinmeden saklanabilir. 1 Byte / bayt olarak yazılıp silinebilir. İhtiyaç duyulduğunda programcı tarafından bu bellek alanına veri yazılabilir. Mikrodenetleyicilerin içerisinde dâhilî olarak EEPROM bellek olabileceği gibi haricî olarak da EEPROM denetleyiciye bağlanabilir.

2.1.2.2. Bellek Organizasyonu Açısından Denetleyici Mimarileri

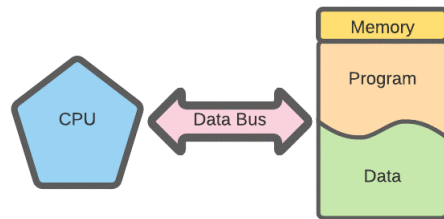
Bellek organizasyonu açısından mikrodenetleyici mimarileri Von Neumann ve Harvard mimarileri olmak üzere ikiye ayrılır.

Bu mimariler ABD savunma bakanlığının askeri bir proje geliştirme amacıyla açtığı bir yarışmada ortaya çıkmıştır. “Von Neumann” mimarisi Princeton Üniversitesi tarafından geliştirilmiştir. Harvard mimarisi de Harvard Üniversitesi tarafından geliştirilmiştir.

Von Neumann Mimarisi

Von Neumann mimarisi ismini ünlü matematikçi John Von Neumann’dan almaktadır. John Von Neumann II. Dünya Savaşında ABD’nin Japonya’ya attığı atom bombasını geliştirdiği Manhattan Projesinde yer almıştır. Ayrıca günümüzde kullandığımız bilgisayarların temel mimarisinin geliştirilmesinde rol oynamıştır.

İlk bilgisayarlar, veriyi ve komutları üzerinde deliklerin bulunduğu uzun şerit kağıtlardan almaktaydı. Bu durum fiziksel bir veri girişi olması nedeniyle işlemlerin yavaş işlemesine / ilerlemesine neden olmaktaydı. Bu soruna çözüm bulmak için John Von Neumann, “First Draft of a Report on the EDVAC” adlı eserinde, komutların ve verilerin bilgisayar içerisinde bulunan bir bellekten sağlanması gerektiğini ortaya sunmuştur (Görsel 2.10).



Görsel 2.10: Von Neumann Mimarisi

Von Neumann mimarisinde veri ve program kodları, aynı bellek bloku içerisinde yer alır. Program komutları ve veriler, aynı veri yolu üzerinden iletilir. Her komut çevriminde bir program kodu ya da bir veri hücresine erişim sağlanabilir. Bu sebeple işlem hızı düşüktür. Bu mimarideki işlemciler **CISC [Complex Instruction Set Computer (Karmaşık Komut Kümeli Bilgisayar)]** adı verilir. Bu mimaride bir komutu gerçekleştirmek için çok fazla sayıda dâhilî saat çevrimine ihtiyaç duyulur. Program ve veriler, aynı iletişim yolunu kullandıklarından çok sayıda adresleme yöntemi kullanılmaktadır. Bu mimaride çok sayıda komut tanımı (100'den fazla) bulunur.



BİLGİ

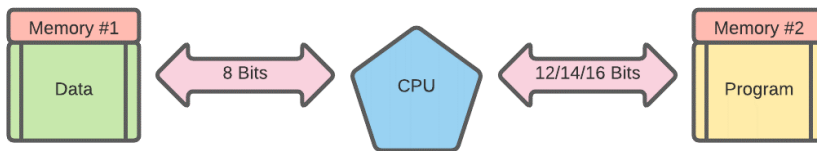
CISC (Complex Instruction Set Computer), karmaşık komut kümeli bilgisayar olarak adlandırılır. Bu mimaride yazılan programların verileri işleyebilmesi için çok fazla komut gerekir. Bu komutları kontrol etmek için gereken kontrol birimlerinin yapısının karmaşık olmasına sebep olur. Farklı komutlar, farklı uzunluklara sahip oldukları için işlem yapılırken işlem hızını düşürür. Bu mimaride kaydedicilerin (registers) az olması ve komutların farklı uzunlukta olması belleklerin daha fazla kullanılmasına yol açar. Çok sayıda komuta sahip olması bazı komutların kullanılmamasına ve bellek israfına sebep olur. Günümüz bilgisayar sistemlerindeki mikro işlemcilerde CISC mimarisi kullanılır.

Program komutları ve veriler, aynı veri yolu üzerinden iletildiğinden veri yolları mikro işlemcinin (CPU) hızına yetişememektedir. Bu durum darboğaza sebep olmaktadır. Darboğazı ortadan kaldırmak için günümüz işlemcilerinde önbellek (cache) yapıları kullanılır. **Önbellek**, işlemci içine yerleştirilmiş küçük bellek bloklarıdır. Bu bellek bloklarına işlenecek olan komut ve veriler, bellekten veri yoluyla getirilerek yerleştirilir. İşlemcinin sürekli ana belleğe giderek veri ve komut alması engellenmiş olur. Önbellek, işlemci performansını arttırmak ve CPU ile bellek arasındaki veri yolu darboğazını engellemek için kullanılmıştır.

Günümüz bilgisayar yapısında Von Neumann mimarisi kullanılır. Yani bilgisayar içerisinde tek bir bellek (RAM) bulunur. Bu bellek alanı hem program komutlarının hem de verilerin tutulduğu bellek bloktur.

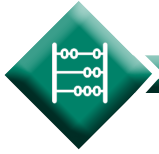
Harvard Mimarisi

Harvard mimarisinde veri (RAM bellek) ve komutların (ROM bellek) bulunduğu bellek blokları birbirinden ayrıdır. Kullanılan iletişim yolları da birbirinden farklıdır (Görsel 2.11).



Görsel 2.11: Harvard Mimarisi

Her bir komut çevriminde hem komut hem de veri iletimi yapılabilir. Bu sayede işlem hızı yükselir. Harvard mimarisine sahip işlemcilere **RISC [Reduced Instruction Set Computer (İndirgenmiş Komut Setli Bilgisayar)]** adı verilir. Bir komut işletilebilmesi için bir dâhilî saat çevrimi yeterlidir. Program ve veri belleği farklı bloklarda olduğundan program yolu ile veri yolu farklı sayıda bit genişliğinde olabilir. Program ve veri belleğinin farklı bloklarda yer alması işlemci dizaynını kolaylaştırır ve program belleği ile veri belleğinin daha verimli kullanılmasını sağlar. Farklı program ve veri bloklarının olması adresleme yöntemlerinin az sayıda olmasını sağlamıştır. Bu mimariye sahip işlemcileri kodlamak için az sayıda komut (50'den az) yeterlidir. Günümüz mikrodenetleyicilerinde Harvard mimarisi kullanılmaktadır.



BİLGİ

RISC (Reduced Instruction Set Computer), indirgenmiş komut kümeli bilgisayar olarak adlandırılır. Bu mimaride komut sayısı azdır. Komutlar CISC mimarisine göre daha basit yapıdadır. Komut sayısının az olması pipeline ve superscalar tekniklerinin kullanılmasına izin verir.

Pipeline: İşlemcinin bir komutu işlerken diğer komutların üzerinde de bazı hazırlık işlemlerini yapabilmesi anlamına gelir.

Superscalar: İşlemcinin her saat sinyalinde birçok komutu okuyarak kendi komut sıralamasına koymasındadır. Bu sayede işlem önceliklerini belirleyerek işlemlerin daha hızlı yürütülmesi sağlanabilir.

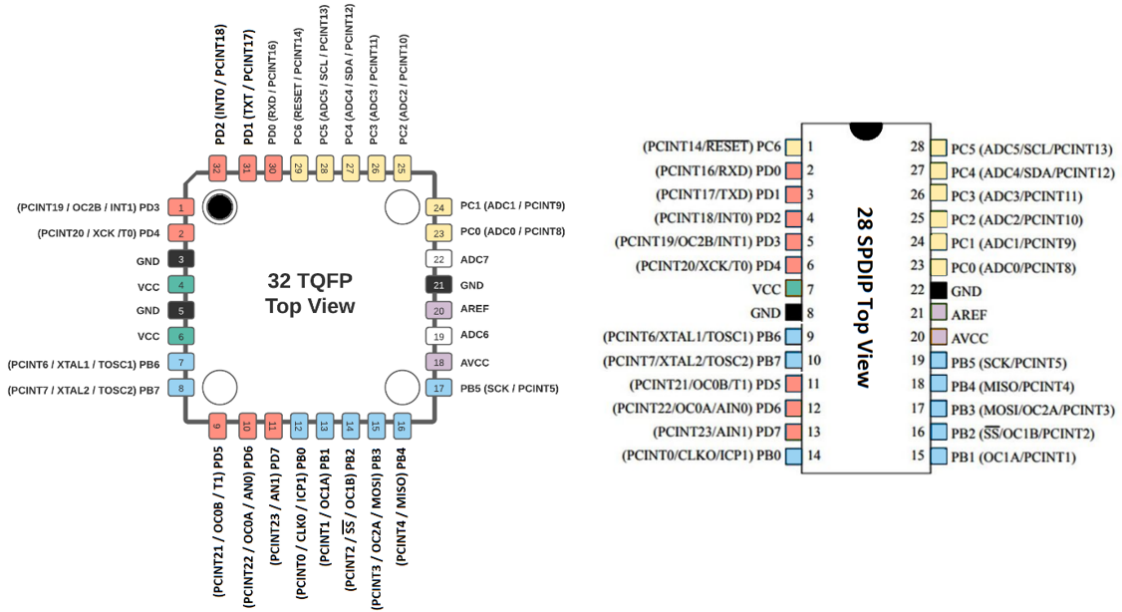
RISC mimarisinin CISC mimarisinden daha hızlı olmasının sebepleri komut sayısının az olması, pipeline ve superscalar tekniğinin kullanılmasıdır. Donanım bakımından basittir. Kaydedici sayısı CISC mimarisine göre daha fazladır. Komutların az olmasından dolayı işletim sistemlerine ve derleyicilere çok iş düşmektedir.

Günümüz mikrodenetleyicilerinde RISC mimarisi kullanılmaktadır.

2.1.2.3. Mikrodenetleyici Pin Diyagramı ve Kılıf Yapıları

Mikrodenetleyicileri bir projede kullanmadan önce denetleyicinin pin diyagramı iyi bilinmelidir. Denetleyicilerin pin diyagramı hakkında en doğru bilgiye, denetleyicinin datasheet sayfalarından ulaşılabilir.

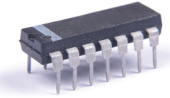
Bu öğrenme birimindeki projelerde kullanılacak mikrodenetleyicinin TQFP ve DIP kılıf tiplerine ait pin diyagramları Görsel 2.12'de verilmiştir.



Görsel 2.12: Atmega328P kılıf tiplerine ait pin diyagramları

Pin diyagramları incelendiğinde pinlerin farklı adlandırıldıkları görülür. Bu isimlendirmeler denetleyici marka, model veya kılıf tipine göre değişiklik gösterebilir. Pin diyagramları incelendiğinde VCC besleme pini, TQFP kılıfında 4 ve 6 No.lu pinler iken DIP kılıfta 7 No.lu pindir.

DIP Kılıf: Kullanımı en kolay kılıf tipi DIP kılıftır. Çift taraflı pinler bulunan bu kılıfın pinleri arasındaki genişlik 0,3 inç ve 0,6 inç olarak üretilir. Bu kılıf türü, delikli montaj teknolojisine uygundur. Açılımı Dual Inline Package'tir. Deney tahtası üzerine montajı kolay olduğundan deneylerde en çok tercih edilen kılıf türüdür (Görsel 2.13).



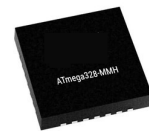
Görsel 2.13: DIP kılıf

TQFP Kılıf: Açılımı Thin Quad Flat Package (Dört Yüzlü İnce Düz Paket)'dir. Yüzey temaslı montaj teknolojisine (SMT) uygun bir paket kılıfıdır (Görsel 2.14).



Görsel 2.14: TQFP kılıf

QFN Kılıf: Açılımı Quad Flat No-Lead'dir. QFN kılıf pin bacaklarına sahip değildir. Pinler, entegrenin dört tarafına dizilmiş bakır noktalardan oluşur. Bacak içermediğinden baskı devre kartı üzerinde fazla yer kaplamaz fakat lehimlemesi diğer kılıflara göre zordur (Görsel 2.15). QFN kılıf daha hassas üretim gerektirmektedir.

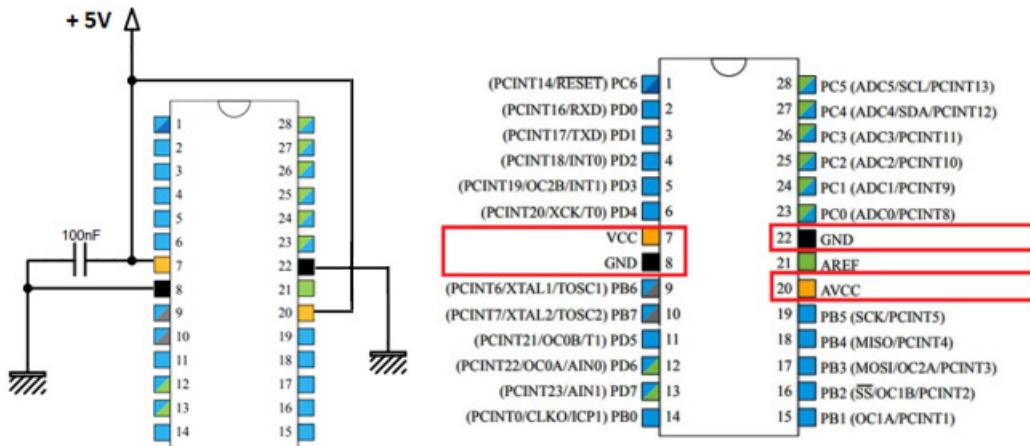


Görsel 2.15: Atmega328P QFN kılıf

Mikrodenetleyiciler daha çok DIP ve TQFP kılıf yapılarında karşımıza çıkar. Bu iki kılıf tipi dışında bazı kılıf tipleri de bulunmaktadır.

2.1.2.4. Besleme Uçları ve Bağlantıları

Besleme uçları, VCC ya da VDD olarak pozitif besleme; VSS ya da GND olarak negatif besleme (yani toprak hattı) olarak ifade edilir. Atmega328P'nin DIP kılıf paketinin pin diyagramı (Görsel 2.16) incelendiğinde 7 No.lu pininin VCC, 20 No.lu pinin AVCC olduğu görülmektedir. Bu pinlere 1,8-5,5 V arası gerilim uygulanmalıdır fakat mikrodenetleyicinin stabil çalışabilmesi için 5 V ideal gerilim değeridir.



Görsel 2.16: Atmega328P besleme uçları ve bağlantıları

Görsel 2.16'da Atmega328/328P mikrodenetleyicisinin besleme uçları ve bağlantısının nasıl yapılacağı verilmiştir. Dikkat edilirse besleme hattı ile toprak hattı arasına 100 nF'lık bir kondansatör bağlanmıştır. Bu kondansatöre **dekuplaj kondansatörü** adı verilir. Bu kondansatör, devreye enerji verildiğinde meydana gelebilecek gerilim dalgalanmalarını önlemek ve oluşabilecek arızaların önüne geçmek için kullanılır.

Atmega328P'nin iki adet besleme bacağı bulunmakta V_{cc} ve AV_{cc} olarak adlandırılır. V_{cc} , dijital besleme gerilimi olarak tanımlanır. AV_{cc} ise Analog-Dijital Dönüştürücü besleme gerilimi olarak tanımlanır. Atmega328P altı adet ADC (Analog-Digital Convert) pini (23 ile 28. pinler arası) bulunmaktadır. Bu pinler dışarıdan alınan analog veriyi dijital veriye dönüştürür. Bu pinlerin doğru sonuçlar üretebilmesi için AV_{cc} pininin, V_{cc} gerilimiyle beslenmesi gerekmektedir. ADC pinleri kullanılmayacaksa bile AV_{cc} pini, V_{cc} gerilimiyle beslenmelidir.

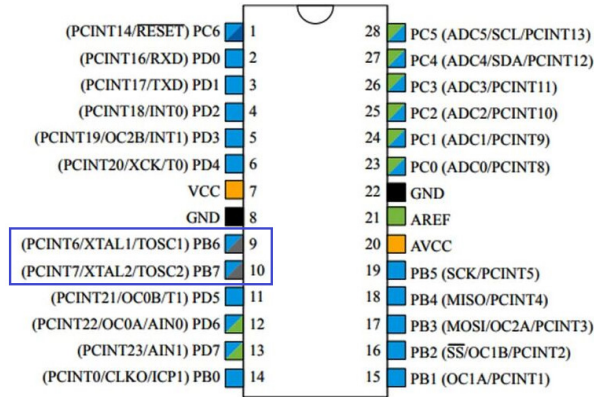
Atmega328P denetleyicisi, güç tüketimi yönünden incelendiğinde 25 °C ve 1,8 V ile aktif modda çalışırken 0,2 mA, güç koruma modunda çalışırken 0,75 uA ve Power-down modunda çalışırken 0,1 uA akım çekmektedir.

Piyasada Atmega328 ve Atmega328P olmak üzere iki farklı model bulunmaktadır. İki denetleyici de aynı mimari yapısına sahiptir. Her iki denetleyiciyi de birbiri yerine kullanılabilir. Bu denetleyiciler arasındaki farklar; Atmega328P denetleyicisinin Atmega328 denetleyicisine göre daha az enerji tüketmesidir. Bunun sebebi ise üretim teknolojilerinden kaynaklanmaktadır. Atmega328P 60 nm teknolojisiyle üretilmişken Atmega328, 90 nm teknolojisiyle üretilmiştir.

2.1.2.5. Osilatör Devreleri

Mikrodenetleyiciler, program belleğinde bulunan kodları yürütmek için kare dalga saat sinyaliye ihtiyaç duyarlar. Saat sinyali üretmek için **osilatör devreleri** kullanılır. Kısaca elektronik devrelerde kare, üçgen, testere vb. sinyalleri üretebilen devrelere osilatör adı verilir.

Atmega328P mikrodenetleyicisinin pin diyagramı incelendiğinde 9. ve 10. pinlerinin OSC (1-2) tanımlamasına sahip olduğu görülmektedir. Bu pinler osilatör devresinin bağlanacağı pinlerdir (Görsel 2.17).



Görsel 2.17: Atmega328P Pin Diyagramı

Bir mikrodenetleyici dört aşamada komut işlemektedir.

1. Aşama: Program sayıcısının (PC) gösterdiği adresten komutun getirilmesi aşamasıdır. Bu işleme **Fetch** (Al-Getir) adı verilir.

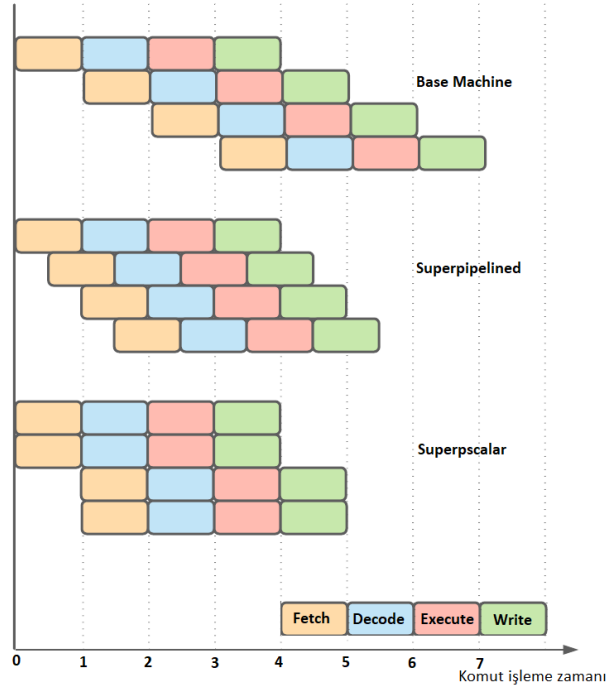
2. Aşama: Kontrol biriminde getirilen kodun çözümlenmesi ve mikro kodların üretilmesi aşamasıdır. Bu işleme **Decode** (Kod Çözümleme) adı verilir.

3. Aşama: Üretilen mikro kodların ALU'da işletilmesi aşamasıdır. Bu işleme **Execute** (Çalıştırma) adı verilir.

4. Aşama: Üretilen sonuçların kaydedicilere yazılması aşamasıdır. Bu işleme ise **Write** (Yaz) adı verilir.

Yukarıdaki işlemler dört saat sinyaliyle işletilir. Bu dört saat çevrimine **bir komut çevrimi** adı verilir. Bir komutun işletilme aşamaları değişmeyeceği gibi bir aşama gerçekleştirilmeden diğer aşamaya da geçilemez.

Denetleyiciler, girişine bağlanan osilatör frekansını dörde bölerek komutları işlemektedir. CISC mimarili işlemcilerde her dört saat çevriminde bir komut işletilmektedir. Bu komut tekniğine **kademeli komut işleme tekniği** adı verilir. Aynı anda yalnızca tek bir komut işletilir (Görsel 2.18).



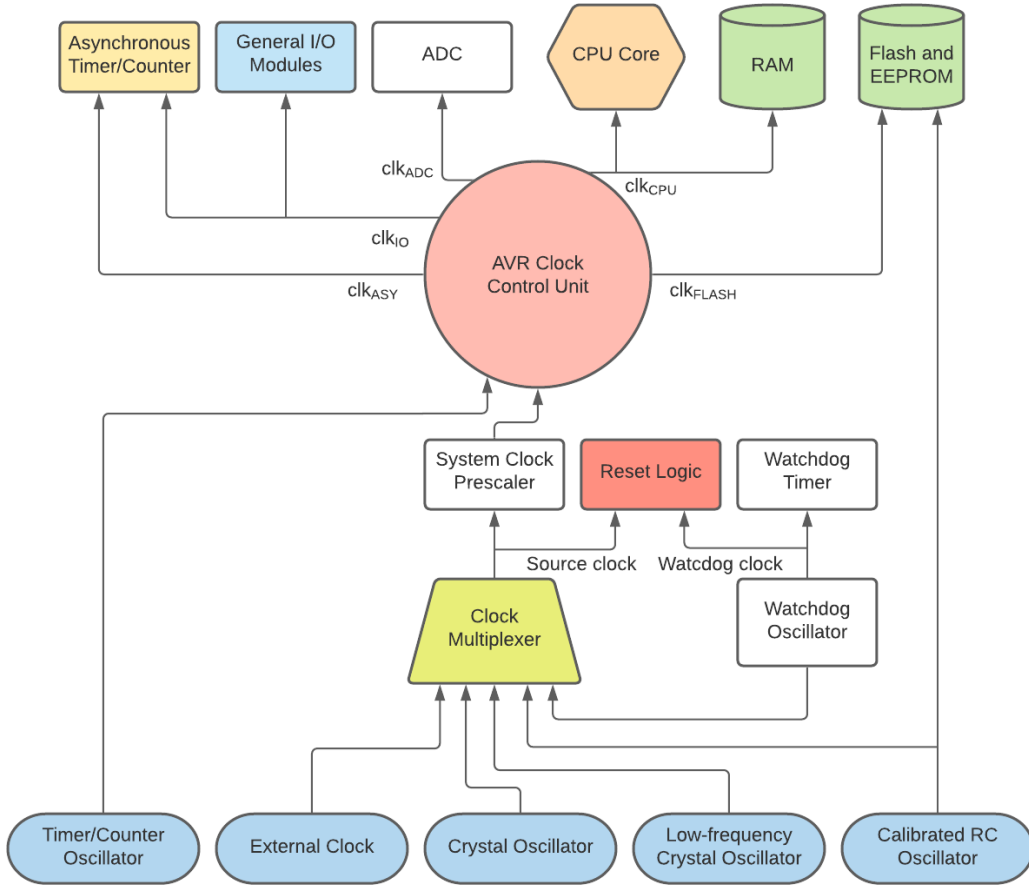
Görsel 2.18: Komut işleme aşamaları

RISC mimarili işlemcilerde performansı artırmak için **pipelining (Kanal Komut İşleme Tekniği)** adı verilen bir teknik kullanılır. Bu teknikte işlemci aynı anda birden fazla komutun işlenmesini sağlar. Örneğin: 1. saat sinyalinde, ilk komut bellekten getirilir. 2. saat sinyalinde, 1. komut çözülürken 2. komut bellekten alınır. 3. saat sinyalinde, 1. komut işletilirken 2. komut çözülür, 3. komut ise bellekten getirilir. CPU birimlerinde her defasında bir komut işlem görmektedir fakat her birim her saat sinyalinde bir işlem gerçekleştirmektedir. Bu işlemler ise farklı komutların farklı aşamalarıdır.

Mikrodenetleyicilere bağlanabilecek osilatör tipleri şunlardır:

- **LP:** Düşük Güçlü Kristal Osilatör (Low Power Crystal Oscillator)
- **FS:** Yüksek Frekanslı Kristal Osilatör (Full Swing Crystal Oscillator)
- **LF:** Düşük Frekanslı Kristal Osilatör (Low Frequency Crystal Oscillator)
- **RC:** Direnç-Kondansatör Osilatör (Resistor/Capacitor)
- **ER:** External Resistor (Haricî Direnç Osilatör)
- **EC:** External Clock (Haricî Saat Girişi)

Görsel 2.19’da Atmega328P denetleyicisinin ana saat sistemi ve kaynakları verilmiştir. Osilatör tipi, mikrodenetleyicinin kontrol edeceği devrenin hız gereksinimlerine göre belirlenir.



Görsel 2.19: AVR denetleyicisi saat sistemi

Kristal Osilatörler ve Bağlantıları

Zamanlamanın önemli olduğu devrelerde kristal osilatörler tercih edilir. Frekans kararlılığı yüksek bir osilatördür yani sabit frekansta kalabilme özelliği yüksektir.

Kristal osilatörlerde salınımı yani kare dalgayı üreten eleman kristal iken seramik osilatörlerde bu eleman seramiktir. Frekans kararlılığının yüksek olması için quartz (kuvars) kristal ya da seramik kullanılır. Görsel 2.20: a ve b’de kristal ve seramik osilatörler verilmiştir.



(a)

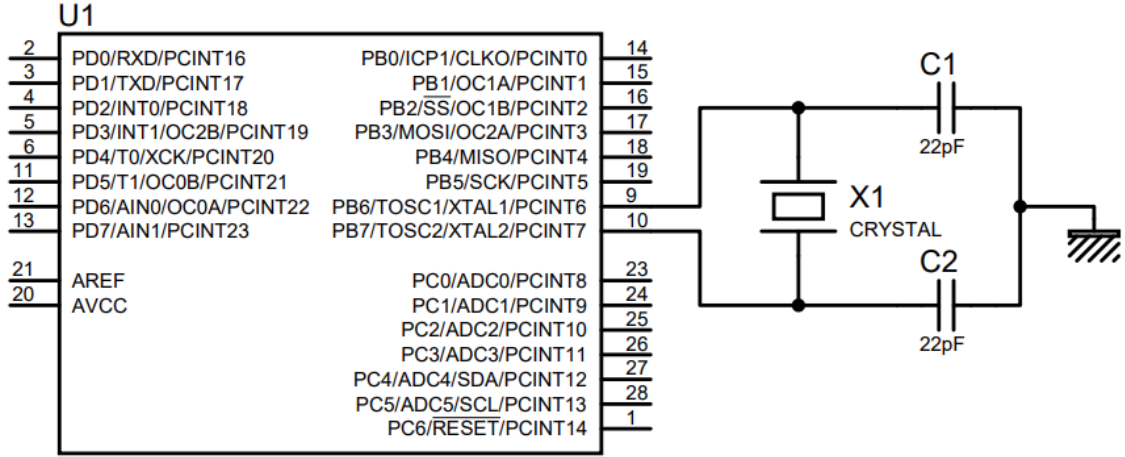


(b)

Görsel 2.20: a) Kristal osilatör

Görsel 2.20: b) Seramik osilatörler

Mikrodenetleyicilerde kristal bağlantısının yapıldığı uçlar, pin diyagramında OSC veya XTAL olarak etiketlenmiştir (Görsel 2.21). Atmega328P denetleyicisinin osilatör bağlantı uçları 9 ve 10 numaralı uçlardır. Bu uçlara kristal osilatör bağlantısı yaparken haricî olarak kondansatör (C1-C2) bağlantısı gerekir (Görsel 2.21). Kondansatörlerin görevi, devreye uygulanan kare dalgadaki değişimleri engelleyerek düzgün dalgalanmayı sağlamaktır. Bu kondansatörlere **dekuplaj kondansatörü** denir.



Görsel 2.21: Mikrodenetleyiciye kristal osilatör bağlama

Kristal osilatör türleri aşağıda verilmiştir.

LP: Düşük Güçlü Kristal Osilatör (Low Power Crystal Oscillator)

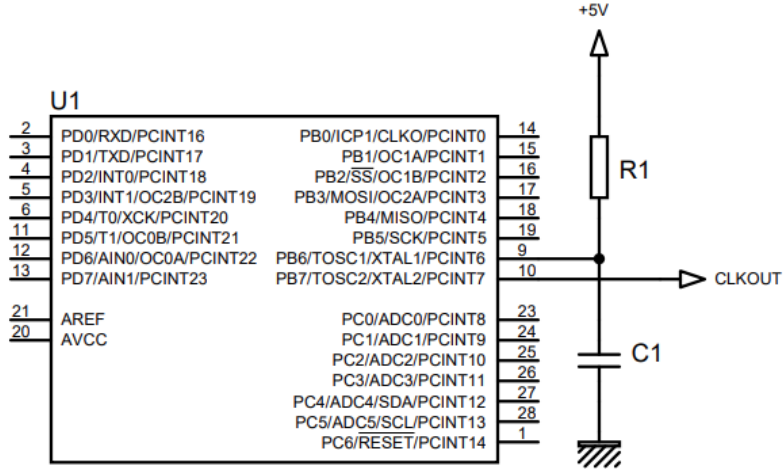
FS: Yüksek Frekanslı (Tam Kapasiteli) Kristal Osilatör (Full Swing Crystal Oscillator)

LF: Düşük Frekanslı Kristal Osilatör (Low Frequency Crystal Oscillator)

Seramik osilatörlerde ise üretim esnasında bu kondansatörler osilatör içine yerleştirilmiştir ve haricî kondansatör bağlantısı gerekmez. Görsel 2.20'de de görüldüğü gibi seramik osilatörler üç bacaklı üretilir. Orta bacak toprak hattına, diğer iki bacak ise denetleyicinin OSC ve XTAL uçlarına bağlanır.

R/C Osilatör ve Bağlantıları

Direnç ve kondansatör kullanılarak oluşturulan bir osilatördür (Görsel 2.22). R/C osilatörleri kristal osilatörlere göre frekans değerlerinde çok fazla sapma yaşanır. Yani hassasiyeti düşüktür. Mikrodenetleyicinin kontrol ettiği zamanlamanın çok önem arz etmediği uygulamalarda tercih edilir. OSC1 (9 No.lu pin) ucuna uygulanan frekans değeri R ve C değerlerine bağlıdır. Belirlenen değerden yaklaşık %10-%20 arası sapma gösterebilir.



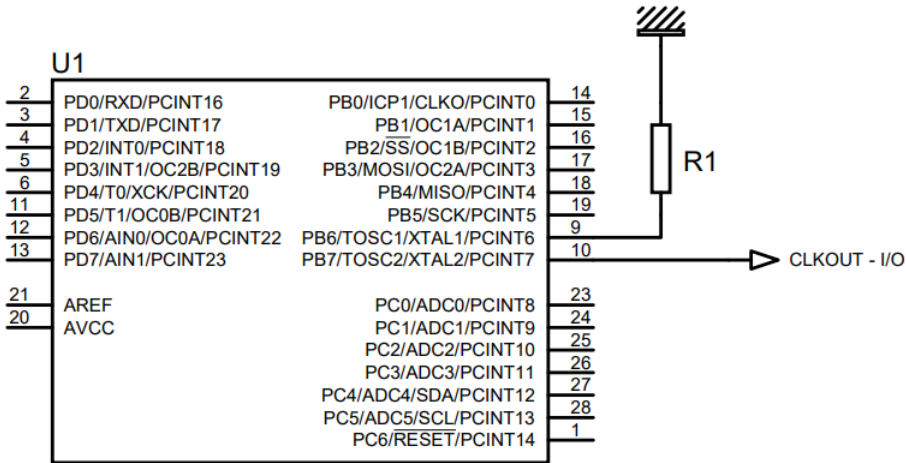
Görsel 2.22: R/C osilatör bağlantısı

Varsayılan olarak dâhilî R/C osilatörü yaklaşık 8.0 MHz saat frekansı sağlar. Bu değer voltaj ve sıcaklık değerlerine bağlıdır. R ve C değerleri değiştirilerek frekans değeri hassas bir şekilde kalibre edilebilir. Görselde 2.22’de verilen CLKOUT ucu mikrodenetleyiciyle eş zamanlı çalışmasını istediğimiz devreler için saat sinyali girişi olarak kullanılabilir. CLKOUT ucundan dâhilî osilatör frekansının 1/8’i alınır. 8 MHz dâhilî osilatör frekansı ile çalışan bir denetleyicinin CLKOUT ucunda 1 MHz frekansında kare dalga alınır.

Ayrıca Atmega328P denetleyicilerinde 128 kHz frekansla çalışan dâhilî R/C osilatörü de mevcuttur.

ER [External Resistor (Haricî Direnç Osilatör)] Bağlantısı

Bu osilatör türünde denetleyiciye yalnız bir adet direnç bağlantısı gerçekleştirilir (Görsel 2.23). Hassasiyeti düşük bir osilatör türüdür. Zamanlamanın önemli olduğu devrelerde tercih edilmemelidir.

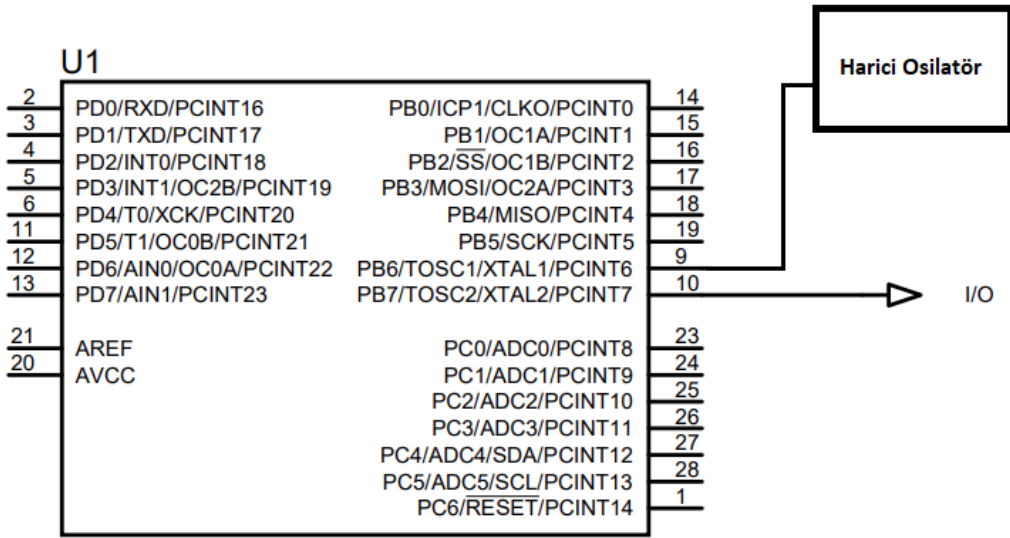


Görsel 2.23: ER osilatör bağlantısı

Denetleyiciye yapılacak konfigürasyonla iki farklı biçimde çalıştırılabilir. İstenirse OSC uçlarından tekine direnç bağlanır, diğer ucundan dâhilî osilatörün 1/8'i oranında sinyal alınır ve eş zamanlı çalışmasını istediğimiz devreye giriş olarak sunulabilir ya da OSC uçlarından tekine direnç bağlanır ve diğer OSC ucu I/O (giriş / çıkış) ucu olarak kullanılabilir.

EC [External Clock (Haricî Saat Girişi)] Bağlantısı

Denetleyiciyi haricî bir saat kaynağından beslemek istendiğinde, Görsel 2.24'te verilen bağlantı kullanılmalıdır. Bu tür bir osilatör kaynağı kullanıldığında OSC2 (10. pin) ucunu I/O ucu olarak kullanabiliriz. Bu osilatör türü zamanlamanın çok hassas olduğu uygulamalarda tercih edilir. Osilatör kaynağı olarak bir bilgisayar ya da programlanabilir entegre osilatör kullanılabilir.



Görsel 2.24: EC osilatör bağlantısı

Haricî bir saat kaynağı uygularken ani frekans değişimlerinden kaçınmak gerekir. Frekans üzerindeki %2'den fazla değişimler, mikrodenetleyici çalışmasını olumsuz etkileyebilir. Frekansta %2'den fazla değişim yapılması gerektiği durumlarda mikrodenetleyicinin kesimde olduğuna emin olunmalıdır.

Denetleyicinin Osilatör Kaynağının Seçimi

Denetleyicinin osilatör ayarları, yazılımla yapılamaz. Denetleyiciye yazılım yüklemesi yapılırken **fuse (sigorta)** adı verilen yazılım yazmaçları ayarlanarak osilatör ayarlamaları gerçekleştirilir. Bunlara **CKSEL** yazmaçları denir. CKSEL yazmacında yapılacak değişiklikler sonucunda saat sistemi, istenilen osilatör kaynağına yönlendirilebilir.

Aşağıda CKSEL yazmaçların durumuna göre hangi osilatör kaynağının aktif olacağını belirten Tablo 2.2’de verilmiştir.

Tablo 2.2: CKSEL Fuse Değerleri

Osilatör Kaynağı	Clock Selector (CKSEL 3..0) Ayarı
Low Power Crystal Oscillator	1111-1000
Full Swing Crystal Oscillator	0111-0110
Low Frequency Crystal Oscillator	0101-0100
Internal 128 kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Mikrodenetleyici ilk kez satın alındığında dâhilî RC osilatörü (8 MHz) kullanılacak şekilde fuse değerleri atanmış (CKSEL = 0010) olarak gelmektedir. Ayrıca 8 MHz’lik saat sinyali sekize bölünmüş işlem hızıyla çalışacak şekilde ayarlanmıştır. $8 \text{ MHz}/8 = 1 \text{ MHz}$ hızla çalışacak şekilde fabrikadan ayarlanarak gelir.

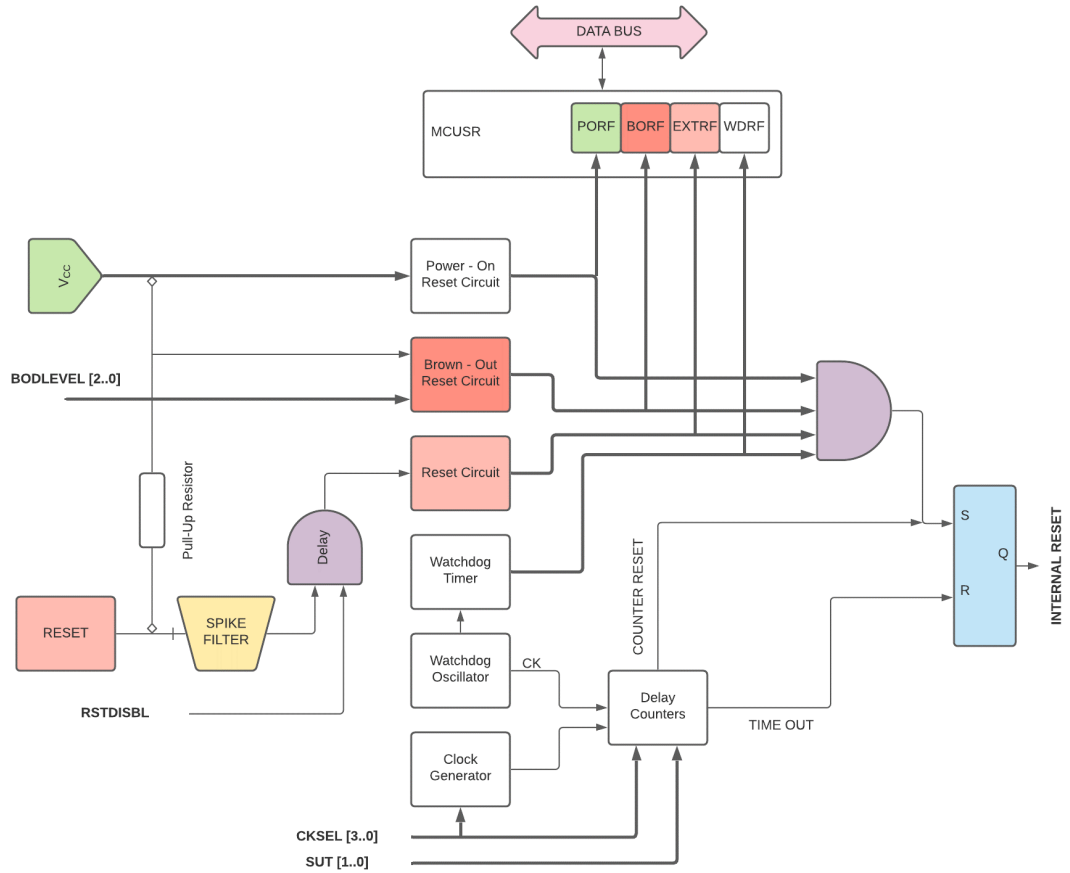
2.1.2.6. Reset (Sıfırlama) Devresi ve Çeşitleri

Bir denetleyicinin resetlenmesi çalışmakta olan programın kesilerek denetleyicinin program belleğindeki başlangıç adresine (0x0000) dönmesi ve programın tekrar çalışmaya zorlanması demektir. Resetleme işlemi hem yazılımsal hem de donanımsal olarak uygulanabilir. Örneğin bir mikrodenetleyicinin RESET (MCLR) ucuna 0 V uygulanarak programın başlangıç noktasından itibaren yeniden başlatılmaya zorlanması donanımsal resettir.

Resetleme, mikrodenetleyicinin kararlı bir şekilde çalışması için uygulanır. Resetleme işleminden sonra bazı kaydedicilerin içeriği sıfırlanırken bazılarının içeriği ise rasgele değişir. Ayrıca tüm I/O kaydedicileri ve ayarları başlangıç değerlerine ayarlanır. Resetleme yapıldığında program belleğindeki komutları sırasıyla okumayı sağlayan PC (Program Counter) içeriği, başlangıç adresini yani 0x0000 konumunu gösterir.

Atmega328P denetleyicisinin dört adet sıfırlama kaynağı vardır (Görsel 2.25).

- Power-On Reset (POR)
- External Reset (RESET)
- Watchdog Timer Reset (WDT)
- Brown-Out Reset (BOR)



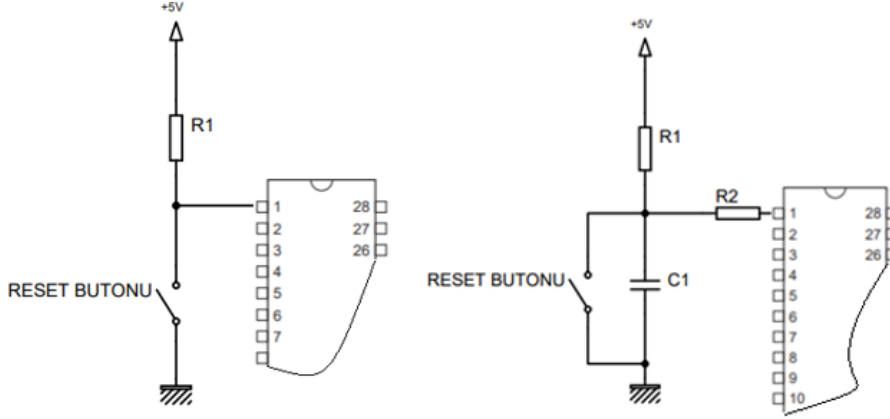
Görsel 2.25: Atmega328P reset devreleri

Power-On Reset (POR)

Mikrodenetleyicinin besleme uçlarına gerilim uygulandığında uygulanan gerilim seviyesinin istenilen düzeye çıkıncaya kadar denetleyiciyi reset konumunda bekletme durumudur. Bu reset durumu gerilim dengeleninceye kadar devam eder. Bu işlemi sağlayan mikrodenetleyici içerisinde dâhilî reset devresi bulunmaktadır (Görsel 2.25).

External Reset [Haricî Reset (RESET-MCLR)]

Mikrodenetleyiciler dışarıdan uygulanan bir sinyalle resetlenebilir. Dışarıdan denetleyicinin RESET(MCLR) pinine lojik 0 sinyali belirli süre uygulandığında denetleyici resetlenir ve çalışmasına program belleğinin 0x0000 adresinden başlar. Normal çalışma esnasında denetleyicinin sürekli resete düşmemesi için reset pininin çeşitli sıfırlama devrelerine bağlanması gerekir.



Görsel 2.26: Haricî reset devreleri [hızlı güç kaynakları için (sol), yavaş güç kaynakları için (sağ)]

Görsel 2.26'daki devreler incelendiğinde reset butonuna basıldığında denetleyicinin reset pini (1 No.lu) 0 V ile yani GND [Toprak (lojik 0)] ile yüklenir. Bu durumda mikrodenetleyicinin PC kaydedicisi 0x0000 adresini işaret eder ve denetleyici program belleğinin en başına döner, tekrar çalışmaya başlar. Reset butonuna basılmadığında ise denetleyicinin 1 No.lu reset pinine güç kaynağı üzerinde +5 V (lojik 1) değeri gelecektir ve resetleme sonlanacaktır.

Denetleyicilerin RESET(MCLR) ucuna bağlı dâhilî bir filtre devresi bulunmaktadır. Bu devre sayesinde RESET ucunda oluşan parazit sinyalleri engellenir ve denetleyicinin kararlı çalışması sağlanmış olur.

Denetleyici beslemesinde VCC gerilimine hızlı ulaşabilen bir güç kaynağı kullanıldığında Görsel 2.26'da soldaki reset devresi kullanılabilir. VCC gerilimine (yani besleme gerilimine) yavaş ulaşan bir güç kaynağıyla besleme yapıldığında sağdaki reset devresi tercih edilmelidir. Bu devrede oluşabilecek parazitler kondansatörle önlenecektir. Bu devrelerdeki R1 dirençleri 4,7 KOhm, R2 direnci 40 KOhm'dan küçük ve C1 kondansatörü 100 nF değerinde tercih edilirse ideal bir reset devresi oluşturulmuş olur.

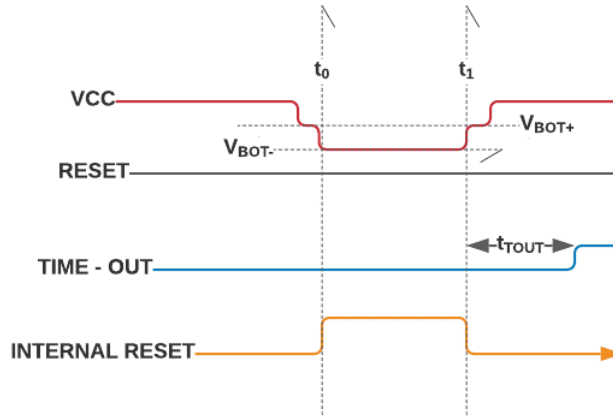
Watchdog Timer Reset (WDT)

Mikrodenetleyicinin normal olarak çalışırken dâhilî bir sayıcıdan (WDT) gelen bir kesme ile resetleme işlemidir. WDT denetleyici içerisinde bulunan dâhilî 128 kHz'lik osilatörün döngülerini sayan bir zamanlayıcıdır. Dâhilî osilatörün her darbesinde WDT sayıcısının değeri bir artırılır.

Mikrodenetleyicilerin işleyebilecekleri komutların maksimum işleme süreleri vardır. Denetleyicinin yapması gereken bir komutu maksimum süre içerisinde yapmaması durumunda; denetleyici sistemde aksaklık olmaması için WDT resetlemesini gerçekleştirir ve denetleyici tekrar çalışmaya başlar. WDT resetlemesi yapılmadan önce denetleyici kritik olan kaydedicilerin içeriğini saklar ve sistemi resetler ve tekrar çalışmaya başlar.

Brown-Out Reset (BOR)

Denetleyici içerisinde bulunan diğer reset devresi Brown-Out Reset (BOR) devresidir. BOR devresi çalışma esnasında VCC değerinde istenilen düzeyin altına düşme gerçekleştiğinde Internal Reset devresini tekrar kurar ve resetleme gerçekleşir. Denetleyiciye uygulanan besleme geriliminin (VCC) belli bir süre (t_{TOUT}), belirli gerilimin (V_{BOT+}) altına düştüğünde denetleyici resetlenir. Atmega328P denetleyicisinde VCC seviyesini izlemek için özel bir devre (BOD) bulunmaktadır (Görsel 2.27).



Görsel 2.27: Brown-Out Reset operasyonu

MCU Status Register [MCUSR (Mikrodenetleyici Durum Kaydedicisi)]

MCU Status Register, hangi reset kaynağının mikrodenetleyiciyi resetlediği hakkında bilgileri tutan bir kaydedicidir. İçerisinde dört adet durum bayrağı (Görsel 2.28) yer almaktadır. Bu bayraklar okunarak sıfırlamanın (reset) hangi kaynaktan geldiği belirlenebilir.

Bit	7	6	5	4	3	2	1	0	Adres
	-	-	-	-	WDRF	BORF	EXTRF	PORF	0x35 (0x55)
Okuma/Yazma	R	R	R	R	R/W	R/W	R/W	R/W	
Varsayılan Değer	0	0	0	0	Kullanılan osilatör kaynağına göre				
MCUSR – MCU Status Register – Mikrodenetleyici Durum Kaydedicisi									

Görsel 2.28: MCU Status Register

7-4 arası bitler, Atmega328P denetleyicisinde kullanılmayan bitlerdir. Bu bit değerleri her zaman sıfır (0) olarak okunacaktır.

Watchdog Timer devresinden reset işlemi gerçekleştiğinde WDRF bitinin (3. bit) içeriği sıfırlanır.

BOR devresinden reset işlemi uygulanırsa BORF bitinin (2. bit) içeriği sıfırlanır. RESET pinine bağlı devreden lojik 0 değeri okunursa yani haricî reset kaynağından resetleme yapılırsa EXTRF biti (1. bit) sıfırlanır. Mikrodenetleyicinin besleme uçlarına gerilim uygulandığında uygulanan gerilim seviyesinin istenilen düzeye çıkıncaya kadar resetleme işlemi gerçekleşir ve PORF biti (0. bit) bu süre içerisinde 0 (sıfır) değerini alır.

2.1.3. Uygun Mikrodenetleyicinin Seçimi

Mikrodenetleyicili bir proje geliştirirken öncelikle projede kullanılacak mikrodenetleyicinin seçimi yapılmalıdır. Projelerde kullanılabilecek birçok mikrodenetleyici marka ve modeli bulunmaktadır. Mikrodenetleyici marka ve modeli seçilirken aşağıda verilen özellikler göz önünde bulundurulmalıdır.

- Maliyet ve temin etme kolaylığı
- Dijital giriş / çıkış uç sayısı
- Analog giriş / çıkış uç sayısı
- Senkron ve asenkron seri iletişim uç sayısı
- Analog karşılaştırıcının var olup olmaması
- Motor ve servo kontrolü için saat sinyali çıkışı
- Haricî kesme ucu sayısı
- Zamanlayıcıyla (timer) kesme yapılıp yapılamayacağı
- Haricî bellek kontrolü yapılıp yapılamayacağı
- Haricî veri yolu (PC, ISA vb.) olup olmadığı
- Program belleği tipi ve kapasitesi (ROM, EPROM, PROM, FLASH ve EEPROM)
- Veri belleği tipi ve kapasitesi (RAM)
- Program belleği üzerinde kod koruması olup olmadığı
- Dâhilî EEPROM olup olmadığı
- Osilatör frekans değeri
- Enerji sarfıyatı
- Kayan nokta hesaplamasının olup olmadığı
- Müşteri hizmetleri ve teknik destek

Mikrodenetleyici seçimi yapılırken maliyeti minimumda tutmaya çalışmanın yanında geliştirilmesi kolay olan ve çok sayıda doküman ve örnek kod sunan marka ve modelleri seçmek daha iyi olacaktır.

2.1.3.1. Atmega328P Denetleyicisinin Özellikleri

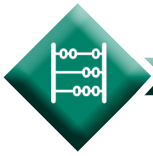
Atmel, 1984 yılında kurulmuş bir yarıiletken üretici firmasıdır. Atmel firması 8051 türevi olan AT91SAM, AT91CAP ve ARM tabanlı mikrodenetleyiciler üretmektedir. Bunun yanında AVR ve AVR32 mimarili mikrodenetleyicileri de üretmektedir. Mikrodenetleyici piyasasına hâkim olan Microchip firması, 2015 yılında Atmel firmasını satın almıştır. Microchip firmasının yanı sıra Philips, Zilog, Parallax, Maxim-Dallas vb. firmalar da mikrodenetleyici üretmektedir.

Microchip firması tarafından üretilen PIC (Peripheral Interface Controller) mikrodenetleyicileri, yıllardır piyasaya hâkim olmuştur. Bu denetleyiciler ekonomik ve kolay ulaşılabilir olmaları sebebiyle denetleyiciler içerisinde en çok kullanılan ürünlerdir. Bu kullanım sıklığı nedeniyle çok fazla kaynak bulunmaktadır.

AVR mikrodenetleyiciler, 8051 iç mimarisiyle üretilmiştir. Atmega328P, AVR mimarisine sahip bir mikrodenetleyicidir. Günümüzde PIC mikrodenetleyicileriyle rekabet içerisindeyler. Son yıllarda AVR mikrodenetleyiciler, adlarından çok bahsettirmeye, büyük ve karmaşık projelerde kullanılmaya başlandı.

Öğrenme biriminde kullanılacak Atmega328/328P mikrodenetleyicisinin özellikleri aşağıdaki gibidir.

- Yüksek performanslı, düşük güç tüketimli 8 bitlik RISC işlemci
- 131 işlemci komutu [Komutların çoğu tek çevrimde (Single Cycle) çalışmaktadır.]
- 32 adet 8 bitlik genel amaçlı kaydediciye sahiptir.
- 20 MHz'e kadar saat hızında çalışabilmektedir.



BİLGİ

Saat hızı, işlemcinin çalışma hızını belirtir. PIC mikrodenetleyicileri bir komutu işlemek için dört, saat çevrimine ihtiyaç duyar. Yani işlem hızı, saat hızının dörtte birine karşılık gelir. AVR mikrodenetleyicilerinde ise işlem hızı genellikle saat hızının 1/1 oranına karşılık gelir.

- 32 KB programlanabilir FLASH yapıda program belleğine sahiptir (10 000 okuma-yazma kapasitesi).
- 1 KB EEPROM belleği bulunmaktadır (100 000 okuma-yazma kapasitesi).

- 2 KB dâhilî SRAM yapıda veri belleğine sahiptir.
- 23 adet programlanabilir giriş ve çıkış ayağı
- Altı adet PWM kanalı
- Altı adet 10 bit ADC (Analog Digital Converter)
- Bir adet programlanabilir seri USART
- İki adet SPI seri arayüzü (Master/Slave)
- Bir adet I2C arayüzü
- Bir adet analog karşılaştırıcı
- Programlanabilir WDT
- Pine bağlı uyandırma ve kesme
- Dâhilî ayarlanabilir osilatör devresi
- Haricî ve dâhilî kesme kaynakları
- Altı adet uyku modu
- Açılışta reset ve kararma (Brown-Out) saptamasına sahiptir.
- Gerçek zaman sayacı
- İki adet entegre çevrim çarpıcı (Multiplier)
- İki adet 8 bit zamanlayıcı (Sayıcı ölçeklendirme ve karşılaştırma)
- Bir adet 16 bit zamanlayıcı (Sayıcı ölçeklendirme ve karşılaştırma)
- 1,8-5,5 V arası çalışma gerilimi
- -40-105 °C arası çalışma sıcaklığı
- Aktif modda 0.2 mA güç tüketimi
- Power-down modda 0.1µA güç tüketimi
- Power-save modda 0.75µA güç tüketimi
- 85 derecede 20 yıl, 25 derecede 100 yıl veri saklama özelliğine sahiptir.

Atmega328 ve Atmega328P, aynı mimariye sahip denetleyicilerdir. Projelerde, birbirinin yerine herhangi bir sorun olmadan kullanılabilir. Aralarındaki farklar, enerji tüketiminde ortaya çıkar. Datasheet sayfalarına bakıldığında Atmega328, 90 nm ile üretilirken Atmega328P, 60 nm ile üretilmiştir. Bu durum, Atmega328P'nin Atmega328 denetleyicisine oranla daha az enerji tüketimine sahip olmasını sağlar. Atmega328P denetleyicisinin üretim teknolojisi, QFN kılıf tipinde (SMD teknolojisine uygun) üretilmesine olanak tanımıştır. QFN kılıf, daha hassas üretim gerektirir.

2.1.3.2. Mikrodenetleyici Komut Seti

Atmega328P denetleyicisi RISC komut setine sahip, yüksek performanslı bir mikrodenetleyicidir. RISC mimarisine sahip olmasına rağmen 131 adet işlemci komutu bulunur. Bu komutların çoğu tek saat çevriminde çalışır ve yazmaçlar üzerindeki verileri kullanarak işlemleri gerçekleştirir. Bu komutlar, RAM belleğin diğer adreslerine doğrudan erişemez. RAM bellekteki veriler üzerinde işlem yapılacaksa RAM bellekteki veriler önce yazmaçlara alınmalı ve işlemler yapıldıktan sonra çıkan sonuçlar, tekrar RAM belleğe yazılmalıdır.

Assembly diliyle kodlama yapılırken Tablo 2.3'teki komutlar kullanılır.

Tablo 2.3: Assembly Diliyle Kodlama Yapılırken Kullanılan Bazı Komutlar

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	$Rd \leftarrow Rd + Rr$	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	$Rd \leftarrow Rd + Rr + C$	Z,C,N,V,H	1
SUB	Rd, Rr	Subtract two Registers	$Rd \leftarrow Rd - Rr$	Z,C,N,V,H	1
AND	Rd, Rr	Logical AND Registers	$Rd \leftarrow Rd * Rr$	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	$Rd \leftarrow Rd \vee Rr$	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	$Rd \leftarrow Rd \oplus Rr$	Z,N,V	1
NEG	Rd	Two's Complement	$Rd \leftarrow 0x00 - Rd$	Z,C,N,V,H	1
INC	Rd	Increment	$Rd \leftarrow Rd + 1$	Z,N,V	1
CLR	Rd	Clear Registers	$Rd \leftarrow Rd \oplus Rd$	Z,N,V	1
MUL	Rd, Rr	Multiply Unsigned	$R1:R0 \leftarrow Rd \times Rr$	Z,C	2

Yukarıdaki tabloda aritmetik ve lojik işlem komutları verilmiştir. Tablo incelendiğinde ilk sütun işlemci komutunun kısaltmasını temsil etmektedir. Bu kısaltma kullanılarak Assembly dilinde kodlama yapılmaktadır. **Operation (İşlem)** sütununda işleme giren yazmaçların hangi işleme girdiği ve çıkan sonucun hangi yazmaca kaydedildiği belirtilmiştir.

Örneğin ADD komutu yazmaçlar üzerinde toplama işlemi gerçekleştiren bir işlemci komutudur. Operation sütununa bakıldığında $Rd \leftarrow Rd + Rr$ işlemi yapıldığı görülmektedir. Bu ifadeyle Rd ile Rr yazmaçlarının değerleri toplanarak sonuç Rd yazmacına yazılır.

Durum kaydedicisi son yapılan aritmetik işlem sonucunda oluşan değere göre çıkan sonucun durumunu gösteren 8 bitlik bir kaydedicidir. Bu kaydedicide bulunan her bir bite bayrak (flag) adı verilir (Görsel 2.29).

Bit	7	6	5	4	3	2	1	0	Adres
Bayraklar	I	T	H	S	V	N	Z	C	
Okuma/Yazma	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	0x3F
Varsayılan Değer	0	0	0	0	0	0	0	0	(0x5F)
SREG – Status Register – Durum Kaydedicisi									

Görsel 2.29: SREG [Status Register (Durum Kaydedicisi)]

Tablo 2.3'te yer alan **Flags** sütunu, komutun işletilmesi sonucu durum kaydedicisindeki hangi bayrakları etkilediğini göstermektedir.

Örneğin MUL komutu işletildiğinde Z [Zero (Sıfır)] ve C [Carry (Elde)] bayrakları etkilenmektedir.

Atmega328P denetleyicisinin komut setinde bulunan komutların büyük çoğunluğu bir saat darbesinde işletilmektedir fakat bazı komutlar birden fazla saat darbesinde işletilmektedir. Tabloda yer alan **Clocks** sütunu belirtilen komutun kaç saat darbesinde işletileceğini göstermektedir.

Örneğin ADD komutu bir saat sinyalinde işletilirken MUL komutu ise iki saat sinyalinde işletilmektedir. Bu durum denetleyicinin ADD komutunu MUL komutuna göre daha hızlı yaptığını göstermektedir.

Tablo 2.4'te, denetleyicinin diğer işlemci komutları ve açıklamaları verilmiştir.

Tablo 2.4: Denetleyicinin Bazı Diğer İşlemci Komutları

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRANCH INSTRUCTIONS					
JMP	k	Direct Jump	$PC \leftarrow k$	None	3
CALL	k	Direct Subroutine Call	$PC \leftarrow k$	None	4
RET		Subroutine Return	$PC \leftarrow \text{STACK}$	None	4
CP	Rd, Rr	Compare	$Rd - Rr$	Z, N, V, C, H	1
CPC	Rd, Rr	Compare with Carry	$Rd - Rr - C$	Z, N, V, C, H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	If $(Rr(b)=0)$ $PC \leftarrow PC+2$ or 3	None	1/2/3
BREQ	k	Branch if Equal	If $(Z=1)$ then $PC \leftarrow PC+k+1$	None	1/2
BRIE	k	Branch if Interrupt Enabled	If $(I=1)$ then $PC \leftarrow k+1$	None	1/2
BRID	k	Branch if Interrupt Disabled	If $(I=0)$ then $PC \leftarrow k+1$	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P, b	Set Bit in I/O Register	$I/O(P,b) \leftarrow 1$	None	2
CBI	P, b	Clear Bit in I/O Register	$I/O(P,b) \leftarrow 0$	None	2
LSL	Rd	Logical Shift Left	$Rd(n+1) \leftarrow Rd(n), Rd(0) \leftarrow 0$	Z, C, N, V	1
LSR	Rd	Logical Shift Right	$Rd(n) \leftarrow Rd(n+1), Rd(7) \leftarrow 0$	Z, C, N, V	1
BSET	s	Flag Set	$SREG(s) \leftarrow 1$	SREG(s)	1
BCLR	s	Flag Clear	$SREG(s) \leftarrow 0$	SREG(s)	1
SEC		Set Carry	$C \leftarrow 1$	C	1
CLC		Clear Carry	$C \leftarrow 0$	C	1
SEZ		Set Zero Flag	$N \leftarrow 1$	N	1
SET		Set T in SREG	$T \leftarrow 1$	T	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	$Rd \leftarrow Rr$	None	1
LDI	Rd, K	Load Immediate	$Rd \leftarrow K$	None	1
LDS	Rd, k	Load Direct from SRAM	$Rd \leftarrow (k)$	None	2
ST	X, Rr	Store Indirect	$(X) \leftarrow Rr$	None	2
STD	Y+q, Rr	Store Indirect with Displacement	$(Y+q) \leftarrow Rr$	None	2
STS	k, Rr	Store Direct from SRAM	$(k) \leftarrow Rr$	None	2
LPM		Load Program Memory	$R0 \leftarrow (Z)$	None	3
IN	Rd, P	In Port	$Rd \leftarrow P$	None	1
OUT	P, Rr	Out Port	$P \leftarrow Rr$	None	1
PUSH	Rr	Push Register on Stack	$STACK \leftarrow Rr$	None	2
POP	Rd	Pop Register from Stack	$Rd \leftarrow STACK$	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	see specific descr. for sleep function	None	1
WDR		Watchdog Reset	see specific descr. for WDR/timer	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Assembly, düşük seviyeli programlama dili olarak kabul edilir ve makine diline en yakın programlama dilidir. Assembly diliyle program yazmak yüksek seviyeli dillerle program yazmaktan oldukça zordur.



SIRA SİZDE

Atmega328P mikrodenetleyicisine ait “Aritmetik ve Lojik İşlem Komutlarının” görevlerini araştırınız.



SIRA SİZDE

Atmega328P mikrodenetleyicisine ait “Veri Transfer Komutlarının” görevlerini araştırınız.

2.1.4. Mikrodenetleyici Programlama İçin Gerekenler

Mikrodenetleyicileri programlamak ve uygulamalarda kullanmak için aşağıda sıralanan donanım ve yazılımlara ihtiyaç vardır.

Kişisel Bilgisayar: Mikrodenetleyicinin istenilen görevleri yerine getirebilmesi için programlanması gerekmektedir. Bunun için çeşitli editörler kullanılarak farklı dillerde (Assembly, C vb.) kodlar yazılabilir. Bu kodlar, derleyiciler sayesinde derlendikten sonra makine diline çevrilerek .hex uzantılı dosyaları oluşturulur. Bu dosyalar bilgisayar aracılığıyla mikrodenetleyicilere yüklenir.

IDE Yazılımı: Denetleyicileri programlamak için oluşturulan programlar, basit bir metin editöründe yazılabilir. Bunun yerine daha gelişmiş olan IDE yazılımları tercih edilir. IDE yazılımları programlama yaparken otomatik kod tamamlama, hata denetimi, derleme, kod okunurluğunu sağlama vb. görevleri yerine getirebilmektedir.

Programlama Kartı: IDE yazılımlar yardımıyla derlenen programlar, makine diline dönüştürülür. Makine diline dönüştürülen kodlar, programlayıcılar yardımıyla denetleyicilere yüklenir. Piyasada mikrodenetleyici marka ve modellerine göre birçok programlama kartı bulunmaktadır.

Program Yükleme Yazılımı: IDE yazılımı ile oluşturulan programlar derlenerek makine diline dönüştürüldükten sonra programlayıcılar vasıtasıyla denetleyiciye yüklenmelidir. Bunun için özel, program yükleme yazılımları kullanılır.

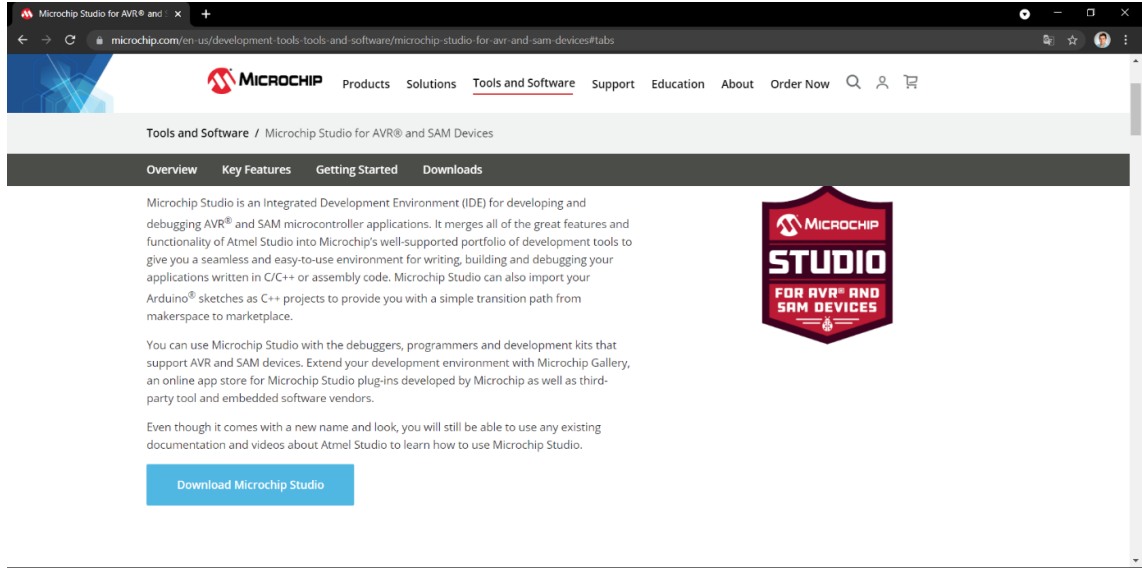
Devre Simülasyon Yazılımları: Bir devrenin modellemesine izin vererek gerçek hayatta karşılaşılabileceğimiz olumsuzlukları önceden denememizi ve çözümler üretmemizi sağlayan yazılımlardır. Bu yazılımlar sayesinde, elektronik devreler bilgisayar ortamında oluşturabilir ve çalışması analiz edilebilir. Mikrodenetleyicili bir elektronik devre oluşturma işleminde, simülasyon yazılımları kullanarak yazılan kodun ve oluşturulan devrenin çalışması önceden analiz edilebilir. Böylece gerçek devre oluşturulmadan önce devrenin analizi yapılarak olası hata ve olumsuzluklar giderilmiş olur.

2.1.5. Programlama Yazılımının Kurulumu

Microchip Studio yazılımı; AVR, UC3 ve SAM mimarisine sahip mikrodenetleyici uygulamalarının geliştirilmesi ve hatalarının ayıklanması için kullanılan IDE yazılımıdır. Kitabımızda Microchip Studio yazılımı kullanılarak projeler oluşturulacaktır.

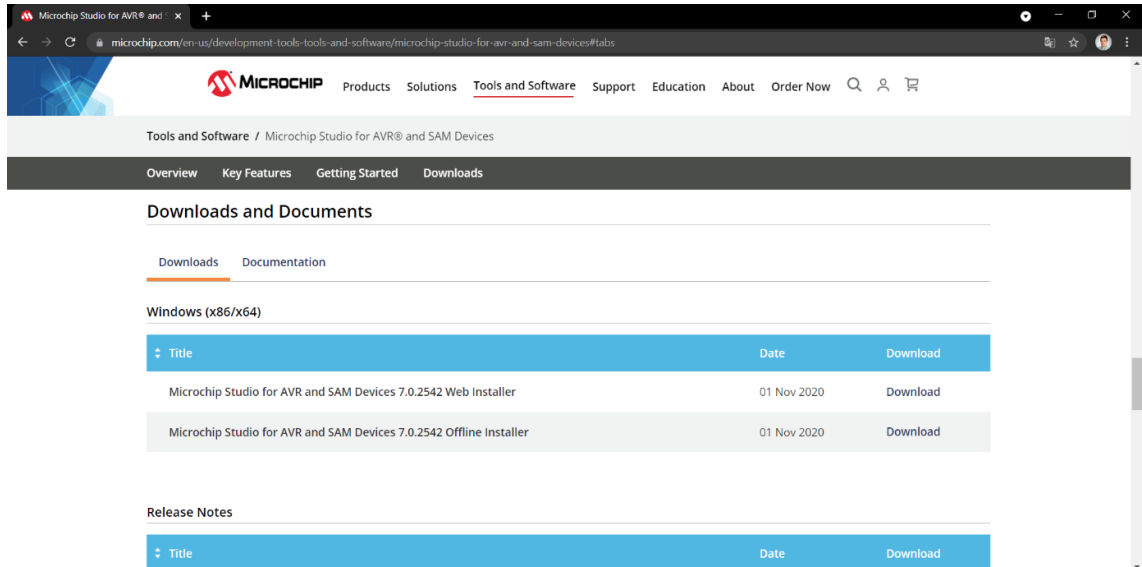
<https://www.microchip.com/en-us/development-tools-tools-and-software/microchip-studio-for-avr-and-sam-devices> adresinden Microchip Studio yazılımının kurulum dosyası indirilebilir.

Yukarıda verilen adrese gidildiğinde Görsel 2.30'daki ekran açılır. **Download Microchip Studio** butonuna tıklayarak Görsel 2.31'deki sayfaya ulaşılır.



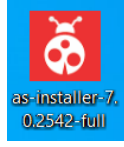
Görsel 2.30: Üretici firma web sayfası

Bu sayfada web tabanlı (Web Installer) ve çevrim dışı (Offline Installer) olarak çalışacak IDE yazılımı kurulum dosyası bağlantıları bulunur. Bu sayfadan “Çevrimdışı Yükleyiciyi” indirmek için **Download** linkine tıklanır (Görsel 2.31).



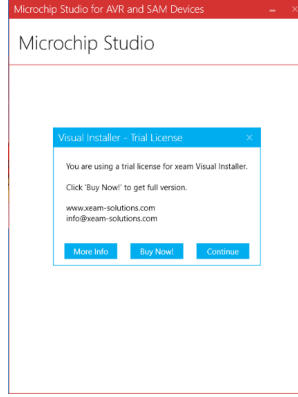
Görsel 2.31: Yazılım indirme sayfası

Yükleme tamamlandığında Görsel 2.32'deki kurulum dosyası bilgisayara indirilir.



Görsel 2.32: Masaüstü kurulum dosyası

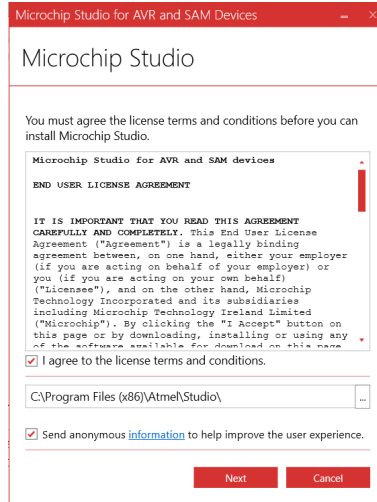
Kurulum dosyasına çift tıklandığında Görsel 2.33'deki ekran gelir.



Görsel 2.33: Versiyon seçim penceresi

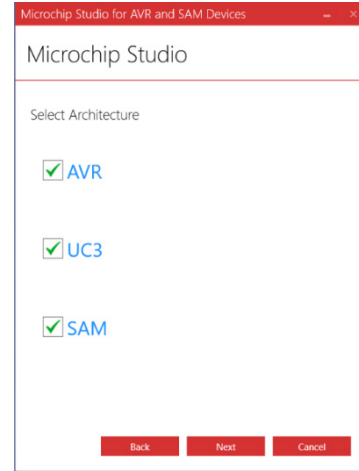
Bu ekranda; Microchip Studio yazılımının hangi versiyonla kurulmak istendiği, yazılımın ücretsiz mi lisanslı mı kullanılmak istendiği sorgulanmaktadır. Lisans satın alınacak ise **Buy Now** butonuna, ücretsiz versiyonu kurmak için **Continue** butonuna basılır (Görsel 2.33).

Görsel 2.34'te Microchip Studio yazılımının lisans sözleşmesi yer alır. Lisans sözleşmesi okunduktan sonra kabul edilirse **Next** butonu aktif olur. Lisans sözleşmesini kabul etmek için **I agree to the license terms and conditions** seçeneği tıklanmalıdır. Bu formda Microchip Studio programının kurulacağı dizin de seçilebilir. Lisans sözleşmesi kabul edildikten sonra Next butonuna basılır.

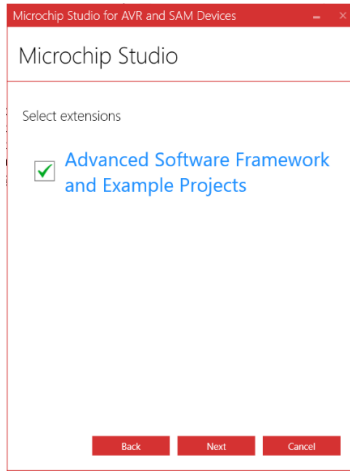


Görsel 2.34: Lisans sözleşmesi ve kurulum dizini seçim penceresi

Görsel 2.35'teki ekrandan denetleyici mimarisi seçimi yapılır. Atmega328 denetleyicisi AVR mimarisine sahiptir. **AVR** mimarisi seçilerek **Next** butonuna basılır. UC3 ve SAM mimarilerine sahip denetleyiciler için de yazılım geliştirilecekse bu seçeneklerin de işaretlenmesi gerekir.



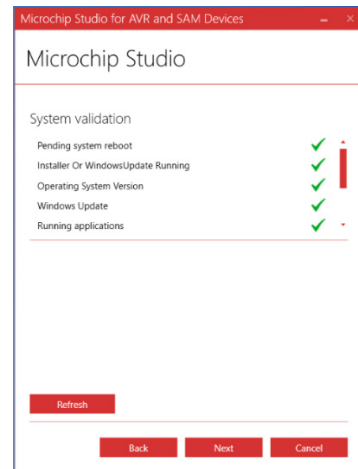
Görsel 2.35: Denetleyici mimarisi seçim penceresi



Görsel 2.36'teki ekranda ise frameworkün son versiyonunu ve örnek projeleri Microchip Studio ile birlikte kurmak isteniyorsa **Advanced Software Framework and Example Projects** seçeneğini aktif ederek **Next** butonuna basılır.

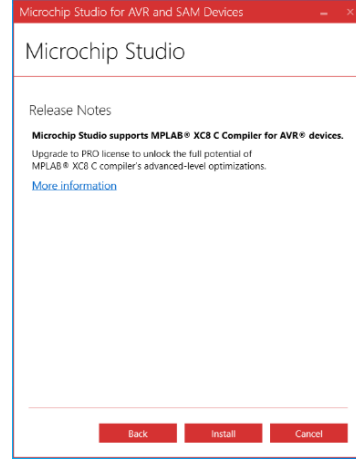
Görsel 2.36: Framework versiyonu ve örnek proje onay penceresi

Görsel 2.37'deki ekranda kişisel bilgisayarlardaki donanımın Microchip Studio yazılımı için uygun olup olmadığı doğrulanır. Doğrulama işlemi; işletim sistemi versiyonu, diskte bulunan boş alan miktarı, ekran kartının yazılım için yeterli olup olmadığı, işletim sistemi güncellemelerinin yapıp yapılmadığı vb. kontrol edilir. Sistem, yazılım için yeterli donanım ve yazılıma sahipse **Next** butonuna basarak bir sonraki adıma geçilir.

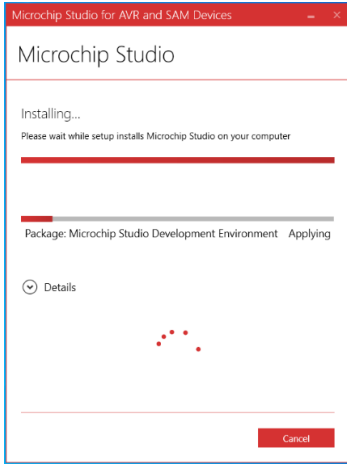


Görsel 2.37: Donanım uygunluk test penceresi

Görsel 2.38'deki **Install** butonuna basarak yazılımın kurulumuna başlanır.

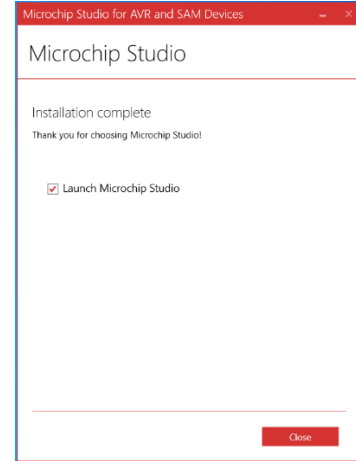


Görsel 2.38: Sürüm notları penceresi



Görsel 2.39: Kurulum ilerleme durum penceresi

Görsel 2.40'taki **Close** butonuna basarak kurulum tamamlanır.



Görsel 2.40: Kurulum tamamlandı penceresi

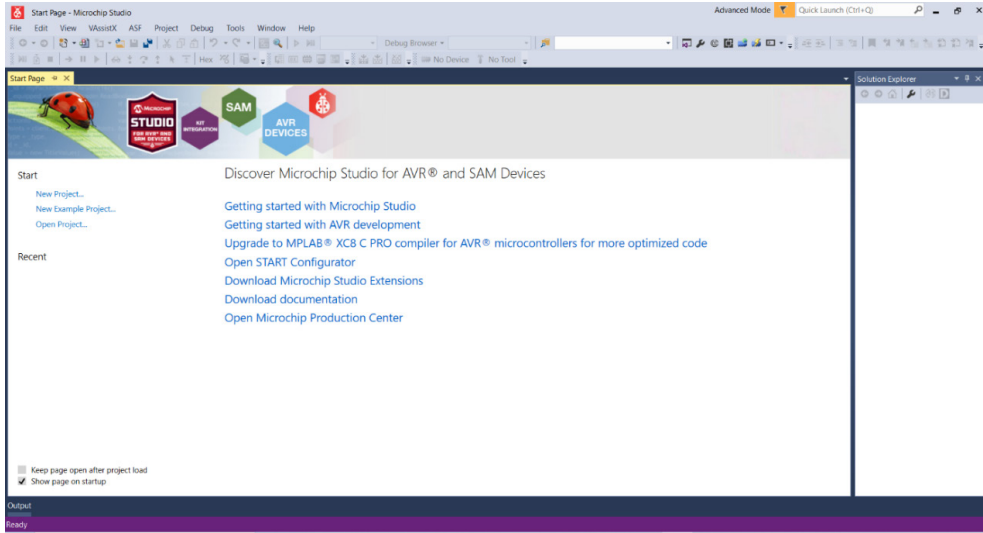


SIRA SİZDE

Web sayfasından Microchip Studio yazılımı kurulum dosyasını bilgisayarınıza indirerek yazılımı kurunuz.

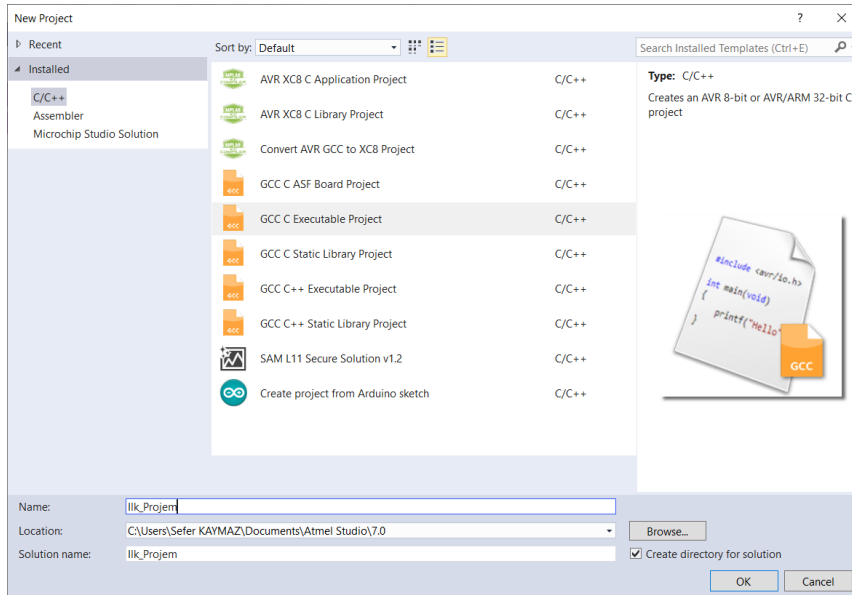
2.1.5.1. Proje Oluşturma

Proje oluşturmak için Microchip Studio yazılımı çalıştırılır. Görsel 2.41'deki **Başlangıç Sayfası (Start Page)** açılır. Bu sayfadan **Start** sekmesi altından **New Project** veya menü çubuğundan **File\New\Project** seçeneğine tıklayarak **New Project** diyalog penceresi açılır.



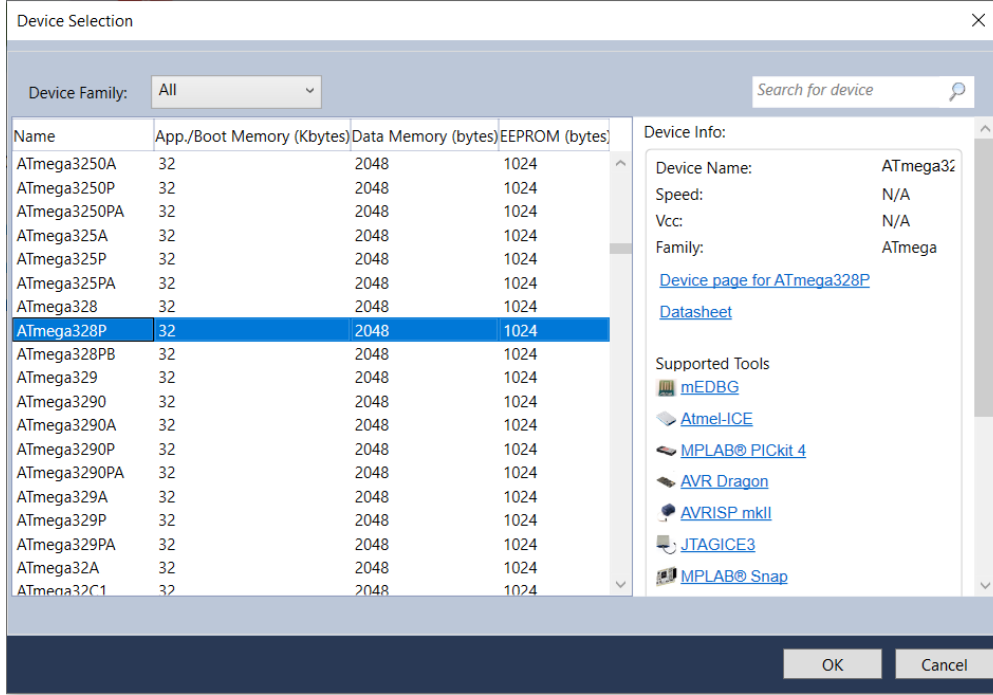
Görsel 2.41: Microchip Studio başlangıç sayfası

Görsel 2.42'de üzerinde çalışılabilecek proje türleri listelenmektedir. Görsel 2.42'deki ekrandan **GCC C Executable Project** seçeneği tıklanır. Bu ekranda, projeye **Name** alanından bir isim verilebilir. **Location** alanından, projenin kaydedileceği dizin seçilebilir. **Browse** butonuna tıklayarak kayıt dizini değiştirilebilir. **Ok** butonuna basarak bir sonraki adıma geçilir.



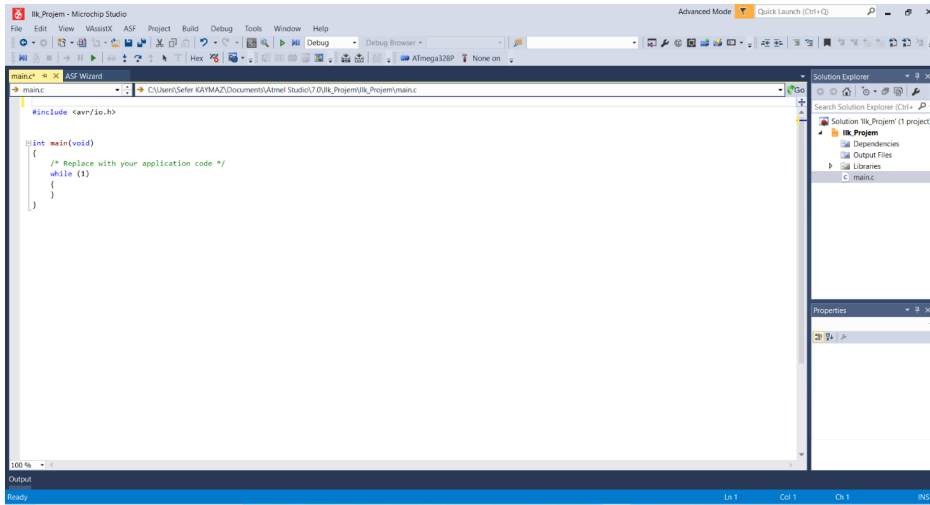
Görsel 2.42: Yeni Proje sayfası

Device Selection penceresinden, programlanacak denetleyici seçimi yapılacaktır. Bu sayfadan mouse yardımıyla liste aşağı-yukarı kaydırılarak **Atmega328P** denetleyicisi seçilir. Bu sayfada kodlanacak aygıtın ismini bulmak için isim arama çubuğu kullanılabilir. Arama çubuğuna 328P yazıldığında Atmega328P denetleyicisi listelenir. Atmega328P denetleyicisi listeden seçildikten sonra **OK** butonuna basılarak proje oluşturma işlemi tamamlanır (Görsel 2.43).



Görsel 2.43: Aygıt (Denetleyici) Seçim Sayfası

Proje oluşturma işlemi tamamlandığında Görsel 2.44'teki ekran görüntüsü gelir. Bu ekranda; kodlama, derleme, hata ayıklama vb. işlemler yapılabilir.



Görsel 2.44: main.c dosyasının içeriği



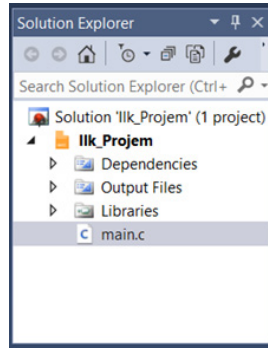
SIRA SİZDE

Microchip Studio yazılımında “ilk_projem” isimli bir GCC C Executable projesi oluşturunuz.

2.1.5.2. Yazılım Arayüz Tanıtımı

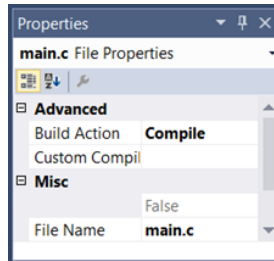
Microchip Studio yazılımının arayüzü aşağıdaki başlıklarda incelenebilir.

Solution Explorer: Microchip Studio ortamında oluşturulan projeye ait dosya ve klasörlerin listelendiği alandır (Görsel 2.45). Bu alanla yeni bir öğe eklenebilir veya eklenmiş öğeler projeden kaldırılabilir. Bu alan kullanılarak öğeler arasında geçiş yapılabilir. Solution Explorer, program arayüzünde görünmüyorsa açmak için “View\Solution Explorer” seçeneği kullanılır ya da klavyeden Ctrl + Alt + L kısayol tuş kombinasyonu kullanılabilir.



Görsel 2.45: Solution Explorer penceresi

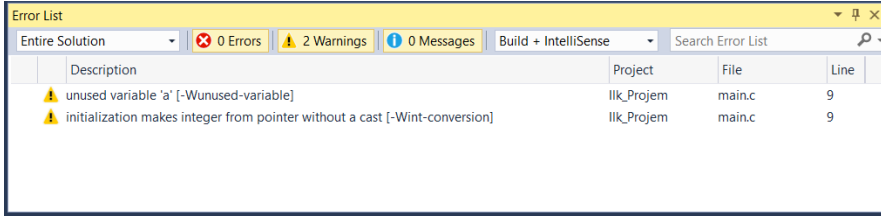
Properties: Bu alan, Solution Explorer araç çubuğunda seçili olan öğeye göre içeriği değişen bir alandır (Görsel 2.46). **Properties** araç çubuğu, Solution Explorer araç çubuğunda seçili olan öğenin özelliklerinin listelendiği ve değişiklikler yapabildiğimiz dinamik bir alandır. Properties, program arayüzünde görünmüyorsa açmak için “View\Properties Window” seçeneği kullanılır ya da klavyeden Alt+Enter kısayol tuş kombinasyonu kullanılabilir.



Görsel 2.46: Properties penceresi

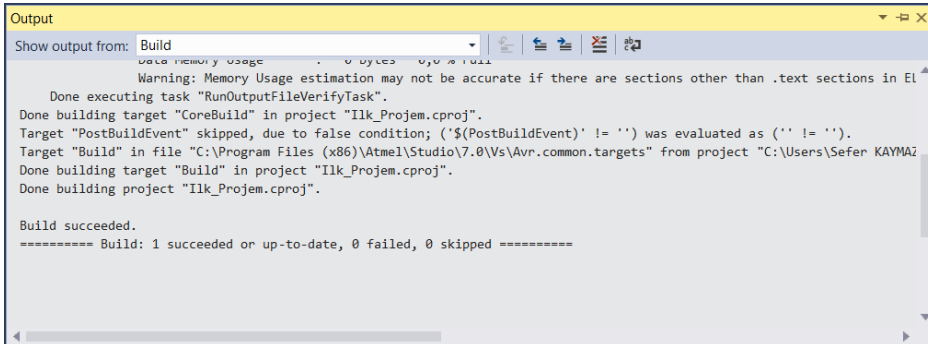
Error List: Projede kodlama esnasında oluşan hata ve uyarıların listelendiği alandır (Görsel 2.47). Kodda bulunan hatalar ve uyarılar satır numarasına göre listelenir. Error List, program arayüzünde görünmüyorsa açmak için “View>Error List” seçeneği kullanılır ya da klavyeden

Alt + Enter kısayol tuş kombinasyonu kullanılabilir.



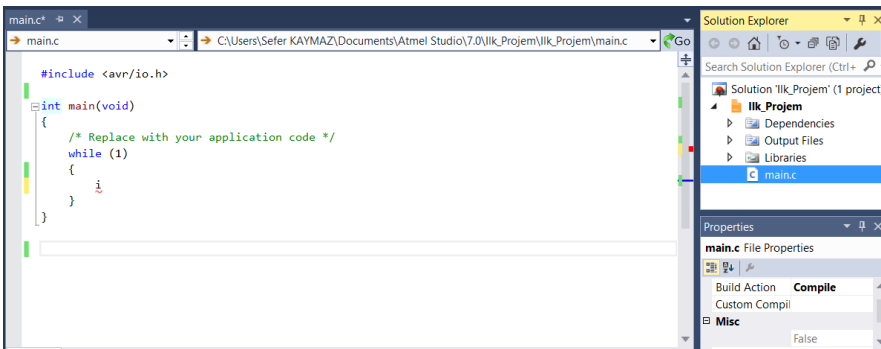
Görsel 2.47: Error List penceresi

Output: C dilinde yazılan kodu, makine diline dönüştürme esnasında yapılan tüm aşamaların sonuçlarının listelendiği çıktı ekranıdır (Görsel 2.48). Derleme esnasında oluşan hatalar ve uyarılar bu ekranda listelenir. Output, program arayüzünde görünmüyorsa açmak için “View\Output” seçeneği kullanılır ya da klavyeden Alt + 2 kısayol tuş kombinasyonu kullanılabilir.



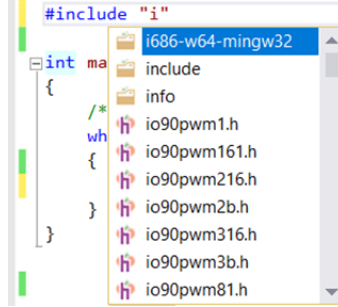
Görsel 2.48: Output penceresi

Kodlar, Görsel 2.49’da görülen **main.c** dosyası içine yazılır. **main.c** dosyası ilk açıldığında içerisinde bir adet main fonksiyonu bulunan bir kod sayfası olarak görüntülenir. Main fonksiyonu ana fonksiyon olarak tanımlanır. Derleyici kodları yürütmeye bu fonksiyondan başlar. Mikrodenetleyiciler, içerisine yüklenen kod bloklarını sürekli yürütmeleri gereken donanımlardır. Main fonksiyonunun içerisinde bir adet while döngüsü bulunmaktadır. Bu döngü aldığı “1” parametresiyle sonsuz döngü biçimini almıştır. Döngü içerisine yazılacak kodlar denetleyicinin enerjisi var olduğu sürece tekrar tekrar yürütülecektir.



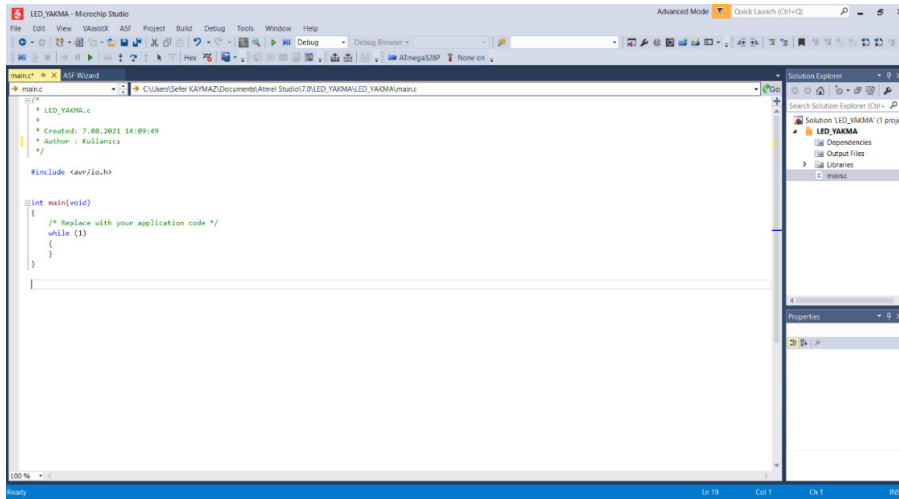
Görsel 2.49: main.c dosya içeriği

Microchip Studio, kod tamamlama özelliğine sahip bir yazılımdır. Yani yazılmak istenen kodun birkaç harfini yazdıktan sonra yazılım, kod seçeneklerini açılır kutu şeklinde sunmaktadır. Açılır kutudan mouse yardımıyla ya da klavye yön tuşlarını kullanarak yazmak istenilen kodun, mouse ile tıklanarak ya da TAB tuşuna basarak seçimi gerçekleştirilebilir (Görsel 2.50).



Görsel 2.50: Kod tamamlama özelliği

Kod Yazma Alanı: Bir proje oluşturulduğunda Görsel 2.51'deki kod yazma penceresi açılır.



Görsel 2.51: Kod yazma penceresi



ÖNEMLİ

Görsel 2.52'deki pencerede /* ... */ sembolleri arasında bulunan ve // sembolleri ile başlayan metinler, **açıklama satırları** olarak adlandırılır. Bu alanlara yazılan yazılar derleyici tarafından dikkate alınmaz ve koda herhangi bir etkide bulunmaz. Programcıya, kodlar arasına notlar alma ve yazılan kodları açıklama imkânı verir.

// sembolleriyle başlayanlar, tek satırlı açıklama satırlarıdır. Alt satıra geçildiğinde kod yazımına devam edilebilir.

/* ... */ sembolleri arasına yazılan tüm veriler, açıklama satırı olarak kabul edilir. /* sembolüyle başlayan satırdan */ sembolüyle biten satıra kadar olan tüm satırlar, açıklama satırı olarak kabul görür.

```
#include <avr/io.h>

int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

Görsel 2.52: Kod yazma alanı

Görsel 2.52’de verilen kod yazma alanı bileşenleri aşağıdaki gibidir.

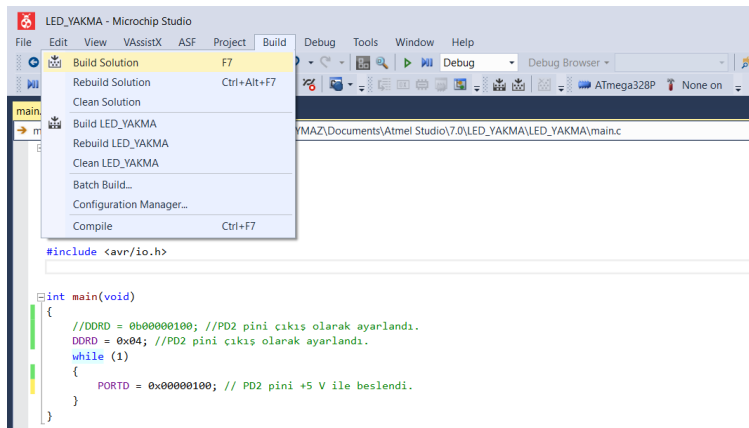
#include <avr/io.h> : Atmega328P mikrodenetleyicisinin portlarını giriş veya çıkış olarak yönetebilmek için projeye, io.h kütüphane dosyasını dâhil etmek gerekir. Atmega328P denetleyicisi AVR mimarisine sahip bir denetleyicidir. Kaynak koda kütüphane dosyalarını ilave etmek için #include yönergesi kullanılır. Microchip Studio programı, proje oluşturulduğunda avr/io.h kütüphane dosyasını koda otomatik olarak dâhil eder.

int main(void) {...} : Bu, ana fonksiyondur. Yazılım, çalışmaya bu fonksiyondan başlar. Bu fonksiyon içerisindeki kodlar, yukarıdan aşağıya doğru bir kez çalışır. Yazılacak ana program, bu fonksiyon içerisine yazılmalıdır.


while (1) {...} : Ana programın sonsuz döngüsüdür. Ana fonksiyon bir defaya mahsus çalışır ve sonlanır. Mikrodenetleyiciler ise verilen görevleri sürekli çalıştıran elemanlardır. Bu yüzden sonsuz döngülere ihtiyaç duyulur. While(1) {...} bloğu arasına yazılan kodlar, denetleyicinin enerjisi kesilinceye kadar sürekli işletilir.

2.1.5.3. Kodun Makine Diline Dönüştürülmesi

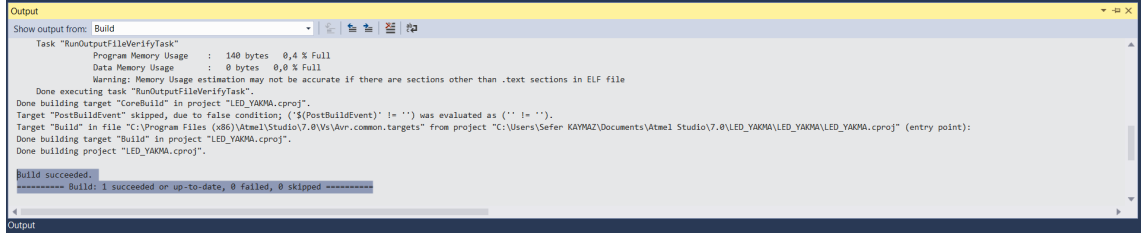
Yazılan kodun makine diline dönüştürülmesi için derlenmesi gerekmektedir. Kodu derlemek için Build menüsü kullanılır (Görsel 2.53).



Görsel 2.53: Kodun derlenmesi

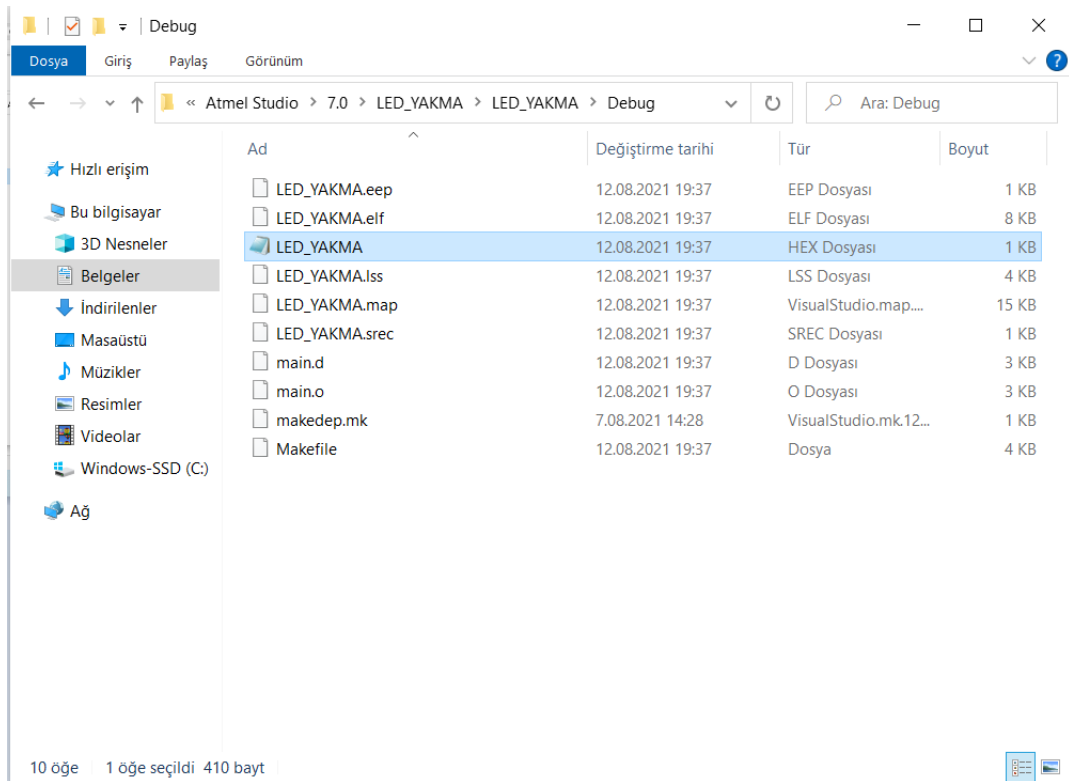
Yazılan kodu derlemek için **Build** menüsündeki **Build Solution** komutu, **F7** kısayol tuşu veya araç çubuğundaki Build () butonu kullanılır.

Buid işlemi gerçekleştirildiğinde Output penceresinde, derleme işleminin adımları ve sonucunda oluşacak mesajlar görüntülenir. Görsel 2.54'te derleme sonucunda oluşan adım ve mesajlar verilmiştir.



Görsel 2.54: Derleme sonucu Output ekranı

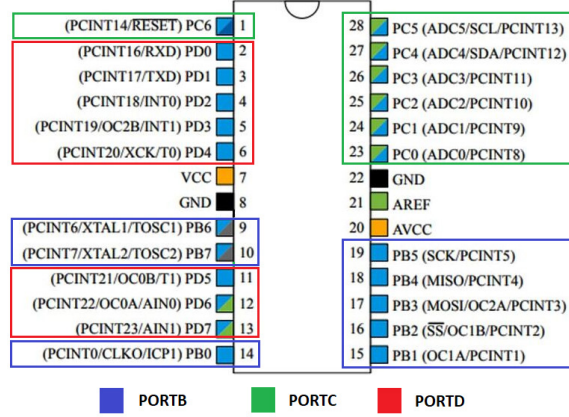
Derleme sonucunda herhangi bir hata oluşmazsa program, makine diline çevrilir ve **hex dosyası** oluşturulur. Makine diline çevrilen kod, hex uzantılı dosyalarda saklanır. Hex dosyası projenin kayıtlı olduğu dizinde, **Debug** klasörünün içerisinde saklanır (Görsel 2.55).



Görsel 2.55: Debug klasörü

2.2. MİKRODENETLEYİCİYLE GİRİŞ-ÇIKIŞ KONTROLÜ

ATMEL Atmega328P mikrodenetleyicisinde üç adet giriş ve çıkış portu bulunur. Bu portlara “Genel Amaçlı Giriş ve Çıkış” [General Purpose Input and Output (GPIO)] adı verilir. Görsel 2.56’da Atmega328P mikrodenetleyicisinin portları ayrıntılı olarak gösterilmiştir.



Görsel 2.56: Atmega328P denetleyicisinin port yapısı

AVR mikrodenetleyicilerinde pinler sekizerli gruplar hâlinde toplanarak portlar oluşturulur. Bu durum, özel kaydedicilerle tüm pinleri tek seferde kontrol etmeye olanak tanır. Her bir pini ayrı ayrı yazılımsal olarak kontrol etmek mümkündür fakat bu durum o pini porttan bağımsız yapmaz. Pinlerin sekizerlik gruplara ayrılmasının en büyük faydası, 7-Segment Display, LCD vb. aygıtların projelerde kullanılırken bu aygıtların veri hatalarını aynı anda ve tek seferde bit değerlerini göndermeye olanak tanımasıdır. Aslında bu pin grupları, paralel iletişim portları olarak kullanılır. Piyasada bulunan birçok dekode ve sürücü entegreleri paralel iletişim protokollerini kullanır. 8 bitlik pin grubuyla 0-255 arası değerler, tek seferde bu portlar üzerinden gönderilebilir ve alınabilir.

Tablo 2.5’te görüleceği gibi GPIO portlarının giriş / çıkış pini olarak kullanılacak pinleri 20 adettir.

Tablo 2.5: GPIO Portlarının Pin Numaraları

PORT	Pin Numaraları	Adet	Açıklama
B	0, 1, 2, 3, 4, 5, 6, 7	6	6-7 No.lu pinleri kristal osilatör bağlantı pinleridir.
C	0, 1, 2, 3, 4, 5, 6	6	6 No.lu pini RESET devresinin bağlandığı pindir.
D	0, 1, 2, 3, 4, 5, 6, 7	8	Tüm pinleri giriş / çıkış olarak kullanılabilir
TOPLAM		20	

Pinlerin giriş (Input) mi çıkış (Output) mı olduğunu ATMEL Atmega328 mikrodenetleyicisi, register (kaydedici) değerleriyle algılar. Mikrodenetleyici içerisinde port durum ve değerlerini tutan kaydediciler bulunur.

GPIO portlarının üç adet ayar kaydedicisi bulunur. Birincisi, pinin giriş mi çıkış mı olduğunu tutan DDRx ayar kaydedicisidir. İkincisi, pinin yönü çıkış olarak ayarlandıysa pinde 0 V (lojik 0) mu

5 V (lojik 1) mu olacağını belirten PORTx kaydedicisidir. Üçüncüsü ise pinin yönü giriş olarak ayarlandıysa pinde 0 V (lojik 0) mu 5 V (lojik 1) mu olduğunu okumayı sağlayan PINx kaydedicisidir. Bunlar özel amaçlı kaydedicilerdir.

Data Memory	
Adres	Bellek
0x0000-0x001F	32 Registers
0x0020-0x005F	64 I/O Registers
0x0060-0x00FF	160 Ext. I/O Registers
0x0100-0x08FF	Internal SRAM (1048 x 8)

Görsel 2.57: SRAM Bellek haritası

Atmega328 mikrodenetleyicisinin 2 kilobayt veri belleği (SRAM) bulunur (Görsel 2.57). Bu belleğin ilk 32 baytlık alanı genel amaçlı kaydedicilere ait iken daha sonraki 64 baytlık (0x0020-0x005F) alan giriş / çıkış portlarının durum ve ayar kayıtlarının saklandığı kaydedicilerdir. Bu alanın 0x0023 – 0x002B adres aralığı DDRx, PORTx ve PINx özel amaçlı kaydedicilerine ayrılmıştır. Bu kaydediciler 8 bitlik hafıza alanlarıdır. Tablo 2.6’da bu kaydedicilerin adres ve bit değerlerinin karşılıkları bulunur.

Tablo 2.6: I/O Kaydedicilerinin Adres ve Bit Değerlerinin Karşılıkları

Adres	İsim	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0
0x14 (0x34)	Rezerve	-	-	-	-	-	-	-	-
0x13 (0x33)	Rezerve	-	-	-	-	-	-	-	-
0x12 (0x32)	Rezerve	-	-	-	-	-	-	-	-
0x11 (0x31)	Rezerve	-	-	-	-	-	-	-	-
0x10 (0x30)	Rezerve	-	-	-	-	-	-	-	-
0x0F (0x2F)	Rezerve	-	-	-	-	-	-	-	-
0x0E (0x2E)	Rezerve	-	-	-	-	-	-	-	-
0x0D (0x2D)	Rezerve	-	-	-	-	-	-	-	-
0x0C (0x2C)	Rezerve	-	-	-	-	-	-	-	-
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x0A (0x2A)	DDRD	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x08 (0x28)	PORTC	-	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x07 (0x27)	DDRC	-	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
0x06 (0x26)	PINC	-	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x04 (0x24)	DDRB	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
0x02 (0x22)	Rezerve	-	-	-	-	-	-	-	-
0x01 (0x21)	Rezerve	-	-	-	-	-	-	-	-
0x00 (0x20)	Rezerve	-	-	-	-	-	-	-	-

2.2.1. DDRx Kaydedicisi

DDRx kaydedicileri denetleyicilerin portlarında bulunan her pinin giriş mi çıkış mı olduğu verisini tutan kaydedicilerdir. Atmega328P denetleyicisinde üç adet DDRx kaydedicisi bulunmaktadır. Bunlar DDRB, DDRC ve DDRD kaydedicileridir. Bu üç kaydedici de sekiz bitlik veri tutabilen kaydedicilerdir. Her bit mikrodenetleyicinin bir pininin giriş-çıkış ayar değerlerini tutar. Kaydedici içerisinde değeri 1 (lojik 1) olan bite karşılık gelen pin çıkış (Output), değeri 0 (lojik 0) olan bite karşılık gelen pin ise giriş (Input) olarak ayarlanmıştır. DDRx kaydedicileri için DDRB, DDRC ve DDRD kaydedicilerin varsayılan değerleri 0 (lojik 0) olarak yani giriş olarak ayarlanmıştır.

DDRD kaydedicisinin içeriği Görsel 2.58’de verildiği gibi ise PD0 ve PD1 pinleri çıkış pini; PD2, PD7 pin aralığı ise giriş pini olarak ayarlanır.

Bit	7	6	5	4	3	2	1	0
DDRD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	0	0	0	0	0	0	1	1
Durum	I	I	I	I	I	I	O	O

- O: Output – Çıkış, I: Input – Giriş

Görsel 2.58: DDRx kaydedicisi

Mikrodenetleyiciler programlamaya başlanmadan önce amaca uygun olarak pinleri giriş mi çıkış mı olacağı ayarlanmalıdır. Bu işlemler DDRx kaydedicilerin değerlerini uygun değerlerle değiştirerek yapılır.

2.2.2. PORTx Kaydedicisi

PORTx (Pin Output Register) kaydedicileri çıkış (Output) olarak ayarlanan pinin 5 V (lojik 1, HIGH) ile mi yoksa 0 V (lojik 0, LOW) ile mi besleneceğinin verisini tutan kaydedicilerdir. Atmega328P mikrodenetleyicisinde üç adet PORTx kaydedicisi bulunur. Bunlar PORTB, PORTC ve PORTD kaydedicileridir. Bu üç kaydedicide sekiz bitlik kaydedicilerdir. Kaydedicilerin her bir biti, denetleyicinin bir pinine karşılık gelir. 1 değerine sahip olan bite karşılık gelen pin, +5 V ile beslenerek ucuna bağlanan cihaza akım verir. 0 değerine sahip olan bite karşılık gelen pin, 0 V ile beslenerek ucuna bağlanan cihaza akım geçişi sunmaz.

DDRD = 0b0000011; // D portunun 0-1 No.lu pinleri çıkış olarak ayarlandı.

PORTD = 0b00000011;

Görsel 2.59’da PORTD kaydedicisinin içeriği verilmiştir. Görsel incelendiğinde PD0 ve PD1 pinlerinde +5 V olduğu diğer pinlerde ise 0 V olduğu görülmektedir.

Bit	7	6	5	4	3	2	1	0
PORTD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	0	0	0	0	0	0	1	1
Volt	0V	0V	0V	0V	0V	0V	+5V	+5V

Görsel 2.59: PORTx kaydedicisi

2.2.3. PINx Kaydedicisi

PINx kaydedicisi, giriş olarak ayarlanan pinlerin +5 V ile mi 0 V ile mi beslendiği verisini tutan kaydedicilerdir. Bu kaydedici içerisindeki veriler okunabilir ve okunan verilere göre çeşitli işlemler yapılabilir. Atmega328P denetleyicisinde üç adet PINx kaydedicisi bulunmaktadır. Bunlar PINB, PINC ve PIND kaydedicileridir. Bu üç kaydedicide 8 bitlik kaydedicilerdir. Kaydedicilerin her bir biti, denetleyicinin bir pinine karşılık gelmektedir. Değeri 1 olan pinin, dışarıdan +5 V ile beslendiği değeri 0 olan pinin ise dışarıdan herhangi bir enerjiyle beslenmediği anlamına gelir.

Görsel 2.60'ta PIND kaydedicisinin içeriği verilmiştir. Görsel incelendiğinde 0, 2 ve 3. bitlerin 1, diğer bitlerin ise 0 değerine sahip oldukları görülmektedir. Bu durumda PD0, PD2 ve PD3 pinleri dışarıdan +5 V ile beslenirken diğerleri 0 V ile beslenmiştir.

Bit	7	6	5	4	3	2	1	0
PIND	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	0	0	0	0	1	1	0	1
Volt	0V	0V	0V	0V	+5V	+5V	0V	+5V

Görsel 2.60: PINx Kaydedicisi



UYGULAMA

Adı:	Mikrodenetleyiciyle LED Kontrolü	No.: 2.1
Amaç:	Mikrodenetleyiciyle LED kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, denetleyicinin dijital pinine bağlı olan LED'i yakan devreyi oluşturunuz ve gömülü yazılımını kodlayınız (Dijital çıkış için D portunun PD2 pinini kullanınız.).

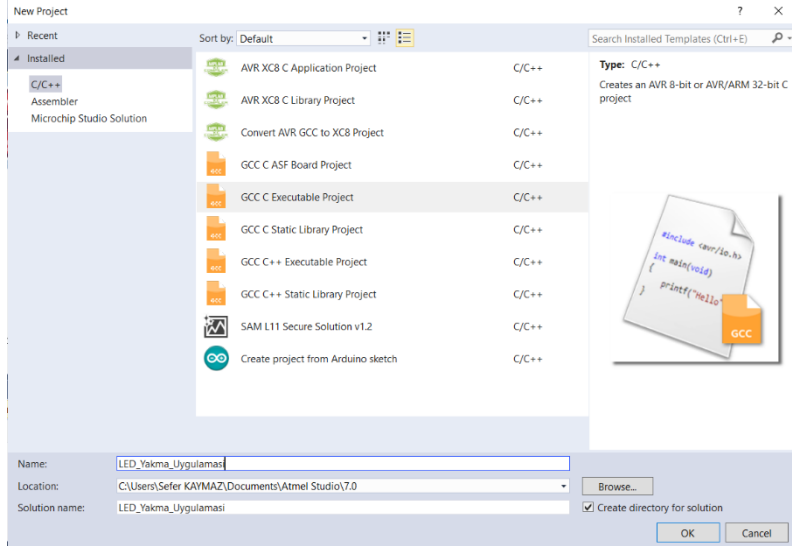
Kullanılacak Araç Gereç: Tablo 2.7'de belirtilmiştir.

Tablo 2.7: 2.1 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı	1 adet
Direnç	220 Ω	1 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	10 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

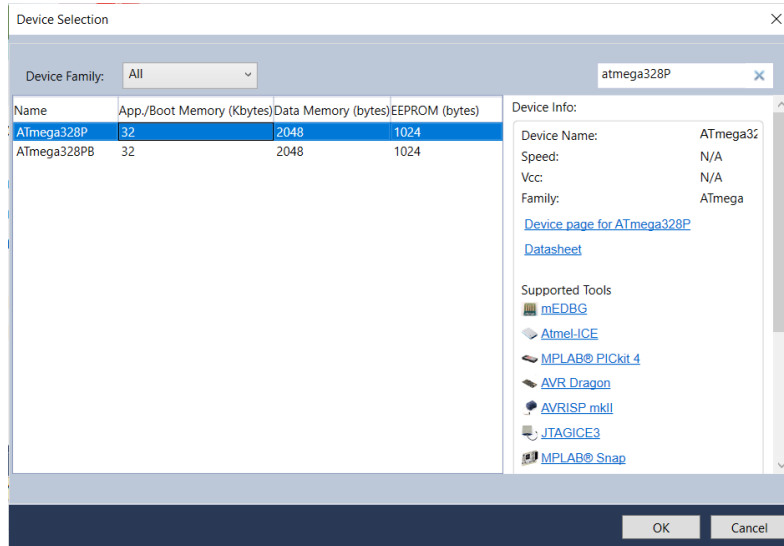
Uygulama Adımları:

1. Adım: Microchip Studio’da File menüsünden New Project seçeneğini seçiniz. Görsel 2.61’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **LED_Yakma_Uygulaması** yazınız.



Görsel 2.61: New Project penceresi

2. Adım: Görsel 2.62’de görülen Device Selection penceresinden Atmega328P mikrodnetleyiciyi seçiniz.



Görsel 2.62: Device Selection penceresi

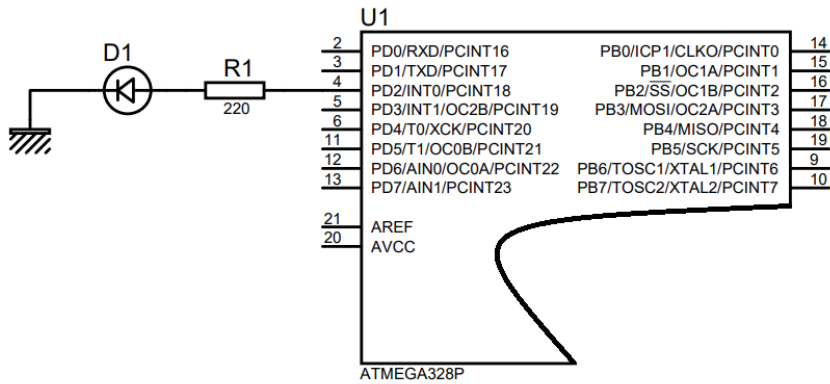
3. Adım: main.c dosyası içine aşağıda verilen kodları yazınız.

```
#include <avr/io.h>

int main(void)
{
    //DDRD = 0b00000100;           //PD2 pini çıkış olarak ayarlandı.
    DDRD = 0x04;                   //PD2 pini çıkış olarak ayarlandı.
    while (1)
    {
        PORTD = 0b00000100;       //PD2 pini +5 V ile beslendi.
    }
}
```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz (HEX dosyası proje klasörünüzde Debug dizini içerisinde bulunur.).

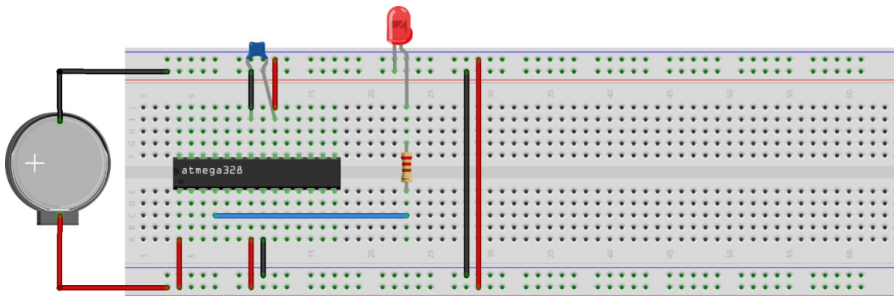
5. Adım: Devre simülasyon programında Görsel 2.63'te verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.63: 2.1 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.64'te verildiği gibi kurunuz.



Görsel 2.64: 2.1 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

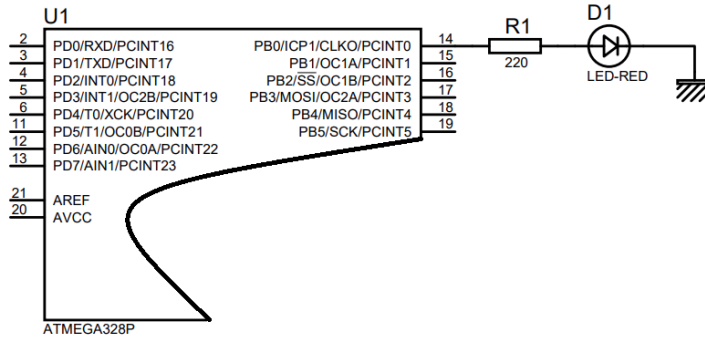
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

Denetleyicinin dijital pinine bağlı olan LED'i yakan devreyi (Görsel 2.65) oluşturunuz ve gömülü yazılımını kodlayınız (Dijital çıkış için B portunun PB0 pinini kullanınız.).



Görsel 2.65: Uygulama devresi

2.2.4. Gecikme İşlemleri

Mikrodenetleyicileri programlarken gecikme fonksiyonları çok fazla kullanılır. Mikrodenetleyiciler, bilgisayarlara göre çok yavaş çalışsa da günlük hayattaki işlemlere göre çok hızlı çalışmaktadır. Program belleğinde yer alan kodları peş peşe ve çok hızlı işler. Günlük hayata uyarlanmış bir problem çözümünde ise zamanı günlük hayata göre ayarlamak yani bazı gecikmeler oluşturmak gerekebilir. Aslında mikrodenetleyicili bir sistemde komutun hızlı çalışması değil, zamanında çalışması önemlidir.

Gecikme esnasında yapılan işlem, denetleyicinin hızlı çalışma gücünü boş bir yerde harcıyıp denetleyiciyi meşgul etmektir. Bu işlem, döngüler yardımıyla gerçekleştirilebileceği gibi denetleyicilerin “Timer” (Zamanlayıcı) devreleri kullanılarak da gerçekleştirilebilir. Döngülerle oluşturulan gecikmelerde, istenilen gecikmenin gerçekleştirilebilmesi için döngü hesabının iyi yapılması gerekir. Bu hesaplamada denetleyicinin çalışma frekansı, önemli rol oynar. Timer’ları kullanarak yapılacak gecikme işlemlerinde ise birçok konfigürasyon ayarı yapılmalıdır. Bunlarla uğraşmak istenmiyorsa C programlama dilinin sunduğu hazır fonksiyonlar kullanılabilir.

Gecikme fonksiyonları projede kullanılacaksa mikrodenetleyicinin bağlı olduğu osilatör frekansını derleyiciye bildirmek gerekir. Derleyici, bu frekans değerini kullanarak gerekli hesaplamaları yapar. Bunun için aşağıda verilen kodu tanımlama alanında bildirmek / belirtmek gerekir.

#define F_CPU Frekans cinsinden osilatör frekansı

#define F_CPU 16000000UL //16MHz’lik osilatör için

#define F_CPU 4000000UL //4MHz’lik osilatör için

Gecikme fonksiyonlarını içeren kütüphane, **delay.h** kütüphanesidir. Gecikme fonksiyonlarını kullanabilmek için delay.h kütüphanesinin projeye dâhil edilmesi gerekir. Bunun için tanımlama alanında aşağıda verilen kodu bildirmek gerekir.

#include <util/delay.h>

Delay kütüphanesi, iki önemli fonksiyon sunar. Bunlar `_delay_ms()` ve `_delay_us()` fonksiyonlarıdır.

`_delay_ms()` Fonksiyonu:

Bu fonksiyon milisaniye cinsinden gecikme sağlar. Parametre olarak 0-65535 arası değer alabilir. Örneğin

- `_delay_ms(100); //100 milisaniyelik,`
- `_delay_ms(1000); // 1 saniyelik (1 000 milisaniye) gecikme sağlar.`

_delay_us() Fonksiyonu:

Bu fonksiyon mikro saniye cinsinden gecikme sağlar. Parametre olarak 0-65535 arası değer alabilir. Örneğin

- _delay_us(100); //100 mikro saniyelik,
- _delay_us(1000); // 1mikro saniyelik gecikme sağlar.

**UYGULAMA**

Adı:	LED Yakma ve Söndürme (Blink Uygulaması)	No.: 2.2
Amaç:	Mikrodenetleyiciyle LED kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, mikrodenetleyicinin PBO pinine bağlı olan LED'i birer saniye aralıklarla yakıp söndüren göz kırpma (blink) uygulama devresini oluşturunuz ve gömülü yazılımını yazınız.

Kullanılacak Araç Gereç: Tablo 2.8'de belirtilmiştir.

Tablo 2.8: 2.2 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı	1 adet
Direnç	220 Ω	2 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	10 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio'da File menüsünden New Project seçeneğini seçiniz. Görsel 2.61'de görülen şekilde, GCC C Executable Project seçeneğini seçtikten sonra Name kısmına **Goz_Kirpma_Uygulaması** yazınız.

2. Adım: Görsel 2.62'de görülen Device Selection penceresinden **Atmega328P** mikrodenetleyiciyi seçiniz.

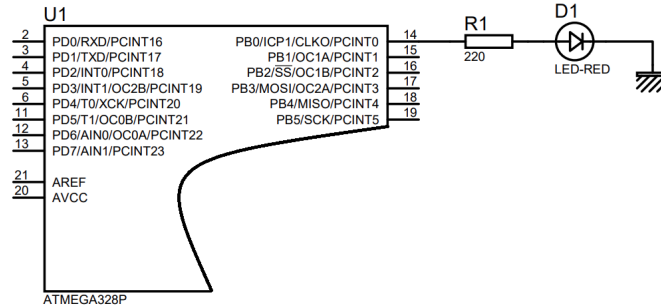
3. Adım: main.c dosyası içine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB = 0x01;                // PB0 pini çıkış olarak ayarlandı.
    while (1)
    {
        PORTB = 0x01;           // PB0 pini HIGH (+5 V) yapıldı (LED yanıyor.).
        _delay_ms(1000);        // Gecikme 1 000 milisaniyedir (1 saniye).
        PORTB = 0x00;           // PB0 pini LOW (0 V) yapıldı (LED sönüyor.).
        _delay_ms(1000);        // Gecikme 1 000 milisaniyedir (1 saniye).
    }
}
```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz.

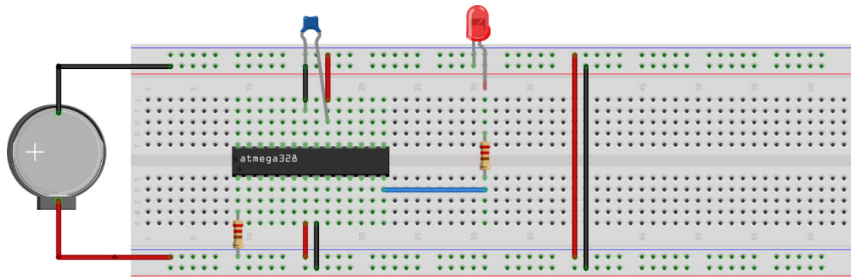
5. Adım: Devre simülasyon programında Görsel 2.66'da verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.66: Uygulama devresi

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.67'de verildiği gibi kurunuz.



Görsel 2.67: 2.2 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

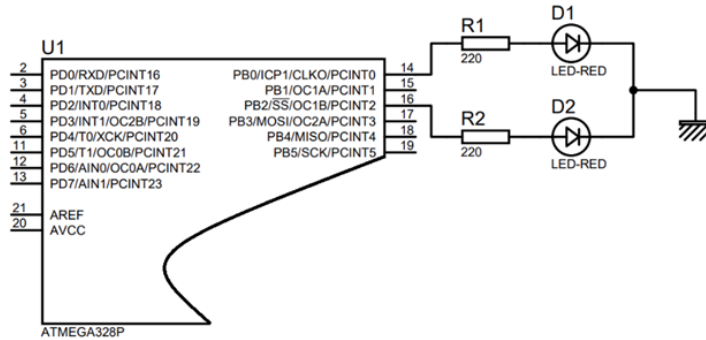
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

Mikrodenetleyicinin PB0 ve PB2 pinlerine bağlı olan LED'leri birer saniye aralıklarla yakıp söndüren uygulama devresini (Görsel 2.68) oluşturunuz ve gömülü yazılımını yazınız.



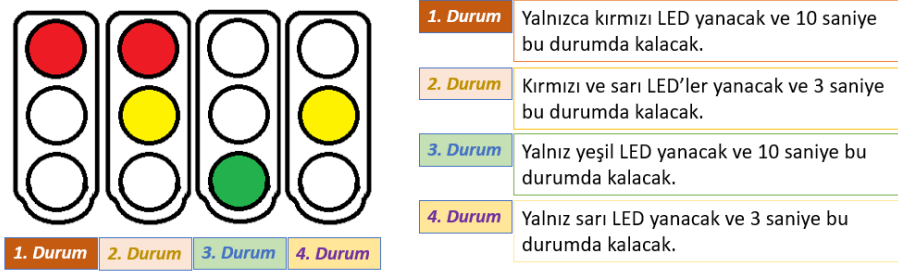
Görsel 2.68: Uygulama devresi



UYGULAMA

Adı:	Trafik Işıkları Simülasyonu	No.: 2.3
Amaç:	Mikrodenetleyici ile Giriş - Çıkış Kontrolü Yapmak	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda kırmızı, sarı ve yeşil LED’lerden oluşan trafik ışıklarının sinyalizasyon devresini oluşturunuz ve gömülü yazılımını Görsel 2.69’daki durum çizelgesine göre kodlayınız (Dijital çıkış için PORTB[0:2] pinlerini kullanınız.).



Görsel 2.69: 2.3 No.lu uygulama için durum çizelgesi

Kullanılacak Araç Gereç: Tablo 2.9’da belirtilmiştir.

Tablo 2.9: 2.3 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı (1), sarı (1), yeşil (1)	3 adet
Direnç	220 Ω	4 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	11 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da File menüsünden New Project seçeneğini seçiniz. Görsel 2.61’de görülen şekilde, GCC C Executable Project seçeneğini seçtikten sonra Name kısmına **Trafik_Isiklari_Uygulamasi** yazınız.

2. Adım: Görsel 2.62’de görülen Device Selection penceresinden **Atmega328P** mikrodenetleyiciyi seçiniz.

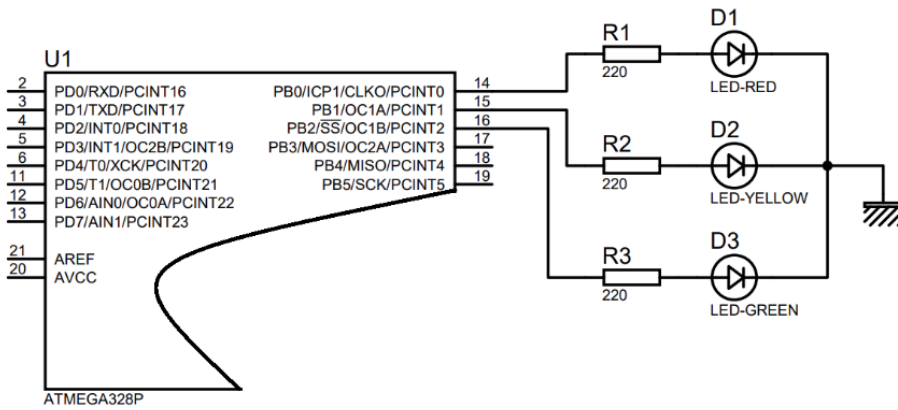
3. Adım: `main.c` dosyası içine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB = 0b00000111;           //PB0, PB1 ve PB2 pinleri çıkış olarak ayarlandı.
    while (1)
    {
        PORTB = 0b00000001;      // PB0 pini HIGH (+5 V) yapıldı (Kırmızı LED yanıyor.).
        _delay_ms(10000);        // Gecikme 10 000 milisaniyedir (10 saniye).
        PORTB = 0b00000011;      // PB0 ve PB1 pinleri HIGH (+5 V) yapıldı
                                   // (Kırmızı ve sarı LED'ler yanıyor.).
        _delay_ms(3000);         // Gecikme 3 000 milisaniyedir (3 saniye).
        PORTB = 0b00000100;      // PB2 pini HIGH (+5 V) yapıldı (Yeşil LED yanıyor.).
        _delay_ms(10000);        // Gecikme 10 000 milisaniyedir (10 saniye).
        PORTB = 0b00000010;      // PB1 pini HIGH (+5 V) yapıldı (Sarı LED yanıyor.).
        _delay_ms(3000);         // Gecikme 3 000 milisaniyedir (3 saniye).
    }
}
```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz.

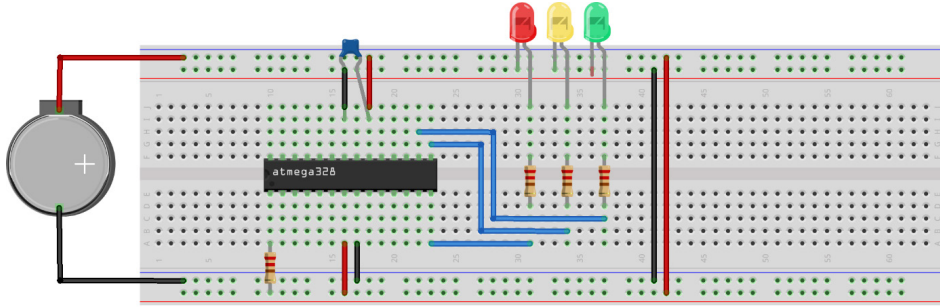
5. Adım: Devre simülasyon programında Görsel 2.70'te verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.70: Uygulama devresi

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.71’de verildiği gibi kurunuz.



Görsel 2.71: 2.3 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

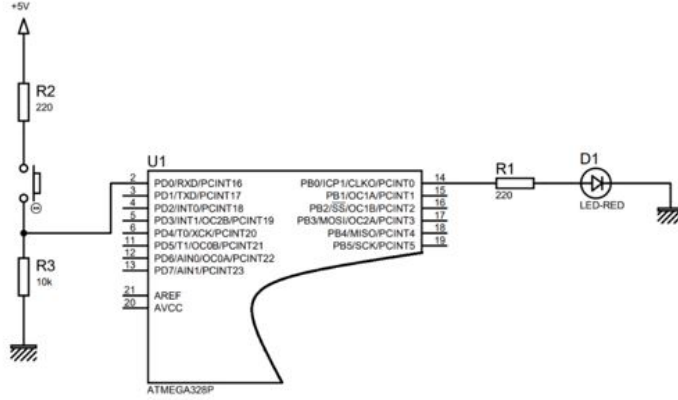
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştırıldı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

PB0 pinine bağlı olan LED'i PD0 pinine bağlı olan butonla yakıp söndüren devreyi (Görsel 2.72) oluşturunuz ve gömülü yazılımını makroları kullanarak kodlayınız.



Görsel 2.72: Uygulama devresi

2.2.5. Bitwise Operatörler

Bir mikrodenetleyici programlamada en önemli konulardan biri de bitisel düzeyde kullanılan operatörlerdir. Yüksek seviyeli bir dille program yazarken fazla tercih edilmeyen bitwise operatörleri, mikrodenetleyici programlamada çok önem arz etmektedir.

C programlama dili, alt seviye özelliklere sahip bir dildir. Yani bit düzeyinde işlemler rahatlıkla gerçekleştirilebilir. Bu alt seviye özellikler sayesinde alt seviye dillere (Assembly vb.) ihtiyaç olmadan birçok gelişmiş program yazılabilir. Günümüzde C dilinin gömülü sistemlerde popülerliğini kaybetmemesinin nedenlerinden biri, alt seviye özelliklere sahip olmasıdır.

Denetleyicinin portlarının kontrolünde bit düzeyinde yapılan değişiklikler önem arz etmektedir. Denetleyicilerin donanımı kontrol edilirken lojik 1 (HIGH) ve lojik 0 (LOW) durumları kullanılmaktadır. Bitwise operatörleri ile 1 ve 0 dönüşümleri çok basit şekilde gerçekleştirilebilmektedir. Bitwise operatörleri sayesinde donanımın kontrolü daha da basitleşmektedir. Tablo 2.10'da bitwise operatörleri verilmiştir. Bu operatörler genellikle programlama dillerinde yer alan mantıksal operatörlerle karıştırılmaktadır. Bitwise operatörleri herhangi bir mantıksal karşılaştırma işlemi yapmamakta bitler üzerinde işlemler gerçekleştirmektedir.

Tablo 2.10: Bitwise Operatörler

Operatör	Açıklama
&	Bitwise AND (AND işlemi)
	Bitwise OR (OR işlemi)
^	Bitwise XOR (Ex-OR işlemi)
<<	Sola Kaydırma işlemi
>>	Sağa Kaydırma işlemi
~	Bit Değil (Tersleme işlemi)

& (AND) Operatörü:

Bu operatör değişken olarak aldığı iki değerın karşılıklı bitlerini **VE** işlemine tabi tutar. AND operatörü mantıksal çarpma işlemi yapar.

ÖRNEK 1:

```
unsigned int sayi1 = 0x9E; //işaretsiz tamsayı olarak sayi1 değişkeni tanımlandı,
                           değışkене 9E heksadesimal değeri atandı.
unsigned int sayi2 = 0x7C; //işaretsiz tamsayı olarak sayi2 değışkeni tanımlandı,
                           değışkене 7C heksadesimal değeri atandı.
unsigned sonuc = sayi1 & sayi2; //sayi1 ve sayi2 değışkenleri bit VE işlemine tabi tutuldu
                              ve çıkan sonuç sonuc değışkenine atandı.
```

sayi1 ve sayi2 değışkenlerinin binary (ikilik) sisteme çevrilmesi:

sayi1 = 1001 1110	sayi2 = 0111 1100
(9) (E)	(7) (C)

sayi1 ve sayi2 değışkenlerinin karşılıklı bitleri üzerinde **VE** işlemi yapılırsa aşağıdaki sonuç elde edilir.

sayi1	1 0 0 1	1 1 1 0	9E
sayi2	0 1 1 1	1 1 0 0	7C
sonuc (sayi1 & sayi2)	0 0 0 1	1 1 0 0	1A

| (OR) Operatörü:

Bu operatör değışken olarak aldığı iki değerin karşılıklı bitlerini **VEYA** işlemine tabi tutar. Or operatörü mantıksal toplama işlemi yapar.

ÖRNEK 2:

```
unsigned int sayi1 = 0x9E; //işaretsiz tamsayı olarak sayi1 değışkeni tanımlandı,
                           değışkене 9E heksadesimal değeri atandı.
unsigned int sayi2 = 0x7C; //işaretsiz tamsayı olarak sayi2 değışkeni tanımlandı,
                           değışkене 7C heksadesimal değeri atandı.
unsigned sonuc = sayi1 | sayi2; //sayi1 ve sayi2 değışkenleri bit VEYA işlemine tabi
                              tutuldu ve çıkan sonuç sonuc değışkenine atandı.
```

sayi1 ve sayi2 değişkenlerinin binary (ikilik) sisteme çevrilmesi:

sayi1 = 1001 1110	sayi2 = 0111 1100
(9) (E)	(7) (C)

sayi1 ve sayi2 değişkenlerinin karşılıklı bitleri üzerinde **VEYA** işlemi yapılırsa aşağıdaki sonuç elde edilir.

sayi1	1	0	0	1	1	1	1	0	9E
sayi2	0	1	1	1	1	1	0	0	7C
sonuc (sayi1 sayi2)	1	1	1	1	1	1	1	0	FE

^ (XOR) Operatörü:

Bu operatör değişken olarak aldığı iki değerın karşılıklı bitlerini XOR (özel veya) işlemine tabi tutar. Özel veya operatörü karşılıklı bitler aynı ise 0, farklı ise 1 değerini döndürür.

ÖRNEK 3:

```
unsigned int sayi1 = 0x9E; //işaretsiz tamsayı olarak sayi1 değişkeni tanımlandı,
                           değışkене 9E heksadesimal değeri atandı.
unsigned int sayi2 = 0x7C; //işaretsiz tamsayı olarak sayi2 değışkeni tanımlandı,
                           değışkене 7C heksadesimal değeri atandı.
unsigned sonuc = sayi1 ^ sayi2; //sayi1 ve sayi2 değışkenleri bit XOR işlemine tabi
                                tutuldu ve çıkan sonuç sonuc değışkenine atandı.
```

sayi1 ve sayi2 değışkenlerinin binary (ikilik) sisteme çevrilmesi:

sayi1 = 1001 1110	sayi2 = 0111 1100
(9) (E)	(7) (C)

sayi1 ve sayi2 değışkenlerinin karşılıklı bitleri üzerinde **XOR** işlemi yapılırsa aşağıdaki sonuç elde edilir.

sayi1	1	0	0	1	1	1	1	0	9E
sayi2	0	1	1	1	1	1	0	0	7C
sonuc (sayi1 ^ sayi2)	1	1	1	0	0	0	1	0	E2

<< (Sola Kaydırma) Operatörü:

Bu operatör, değışkenin bitlerini belirtilen sayı kadar sola öteleme işlemi gerçekleştirir. Kullanımı aşağıda verildiğı gibidir.

değişken << kaydırma sayısı

ÖRNEK 4:

unsigned int sayi = 0xC3; //işaretsiz tamsayı olarak **sayi** değişkeni tanımlandı, değişkene C3 heksadesimal değeri atandı.

unsigned sonuc = sayi << 3; //**sayi** değişkeninin bitleri, verilen sayı kadar (3) sola ötelendi, çıkan sonuç **sonuc** değişkenine atandı.

sayi değişkeninin binary (ikilik) sisteme çevrilmesi:

sayi = 1100 0011

(C) (3)

sayi değişkeninin bitleri üzerinde sola kaydırma işlemi 3 defa yapılırsa aşağıdaki sonuç elde edilir.

sayi	1	1	0	0	0	0	1	1	0xC3
1.kaydırma	1	0	0	0	0	0	1	1	0 ile beslenir.
2.kaydırma	0	0	0	0	0	1	1	0	0 ile beslenir.
3.kaydırma	0	0	0	1	1	0	0	0	0 ile beslenir.
sonuc	0	0	0	1	1	0	0	0	0x18

Sola kaydırma işlemi yapılırken sağdan 0 ile besleme işlemi yapılırken soldaki ilk bit ise kaydırma işlemi sonucunda kaybolur.

>> (Sağa Kaydırma) Operatörü:

Bu operatör değişkenin bitlerini belirtilen sayı kadar sağa öteleme işlemi gerçekleştirir. Kullanımı aşağıda verildiği gibidir.

değişken >> kaydırma sayısı

ÖRNEK 5:

unsigned int sayi = 0xD3; //işaretsiz tamsayı olarak **sayi** değişkeni tanımlandı, değişkene D3 heksadesimal değeri atandı.

unsigned sonuc = sayi >> 3; //**sayi** değişkeninin bitleri verilen sayı kadar (3) sağa ötelendi, çıkan sonuç **sonuc** değişkenine atandı.

sayi değişkeninin binary (ikilik) sisteme çevrilmesi:

sayi = 1101 0011

(D) (3)

sayi değişkeninin bitleri üzerinde **sağa kaydırma** işlemi 3 defa yapılırsa aşağıdaki sonuç elde edilir.

sayi	1	1	0	1	0	0	1	1	0xD3
0 ile beslenir.	0	1	1	0	1	0	0	1	1.kaydırma
0 ile beslenir.	0	0	1	1	0	1	0	0	2.kaydırma
0 ile beslenir.	0	0	0	1	1	0	1	0	3.kaydırma
sonuc	0	0	0	1	1	0	1	0	0x1A

Sağa kaydırma işlemi yapılırken soldan 0 ile besleme işlemi yapılırken sağdaki ilk bit ise kaydırma işlemi sonucunda kaybolur.

~ (Bit Değil) Operatörü:

Bu operatör 1'e tümlleme işlemini gerçekleştirir. 1 olan bitleri 0, 0 olan bitleri ise 1 yapar.

ÖRNEK 6:

```
unsigned int sayi = 0xC3;    //işaretsiz tamsayı olarak sayi değişkeni tanımlandı, değışkene
                             C3 heksadesimal değeri atandı.
unsigned sonuc = ~ sayi;    //sayi değışkeninin bitleri bit değil işlemine tabi tutuldu,
                             çıkan sonuç sonuc değışkenine atandı.
```

sayi değışkeninin binary (ikilik) sisteme çevrilmesi:

sayi = 1100 0011

(C) (3)

sayi değışkeninin bitleri üzerinde bit değil işlemi yapılırsa aşağıdaki sonuç elde edilir.

sayi	1	1	0	1	0	0	1	1	0xC3
sonuc	0	0	1	0	1	1	0	0	0x1C



UYGULAMA

Adı:	Kara Şimşek Uygulaması	No.: 2.4
Amaç:	Bitwise Operatörleri ile Giriş - Çıkış Kontrolü Yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, sekiz adet LED kullanılarak oluşturulmuş Kara Şimşek devresini oluşturunuz ve gömülü yazılımını kodlayınız (Dijital çıkış için PORTD [0:8] pinlerini kullanınız.).

Kullanılacak Araç Gereç: Tablo 2.11’de belirtilmiştir.

Tablo 2.11: 2.4 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı	8 adet
Direnç	220 Ω	8 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	17 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da File menüsünden New Project seçeneğini seçiniz. Görsel 2.61’de görülen şekilde, GCC C Executable Project seçeneğini seçtikten sonra **Name** kısmına **Kara_Simsek_Uygulamasi** yazınız.

2. Adım: Görsel 2.62’de görülen Device Selection penceresinden **Atmega328P** mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRD = 0xFF; // D portunun tüm pinleri çıkış olarak ayarlandı.
    while (1)
    {
        int sola = 0x01; // Soldaki ilk biti bir yapan değişken,
        int saga = 0x80; // Sağdaki ilk biti bir yapan değişkendir.
        for(int i=0; i<8; i++)
        {
            PORTD = sola<<i; // PORTD’nin bitlerini i kadar sola kaydır.
            _delay_ms(100); // 100 milisaniye bekle.
        }
    }
}
```

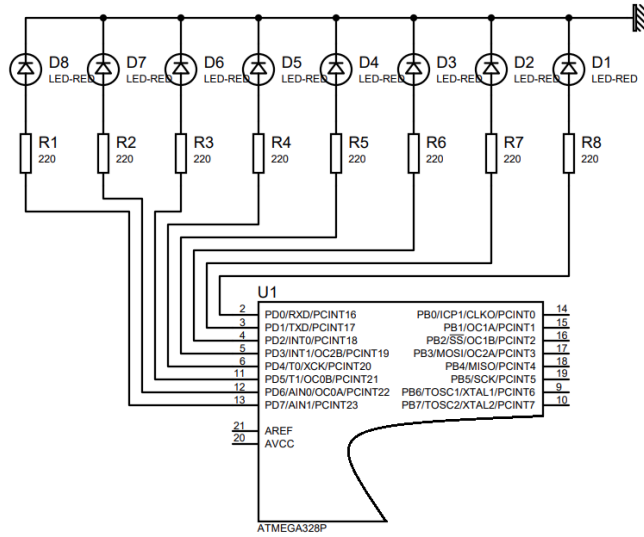
```

for(int i=0;i<8;i++)
{
    PORTD = saga>>i;           // PORTD'nin bitlerini i kadar sağa kaydır.
    _delay_ms(100);           // 100 milisaniye bekle.
}
}

```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz.

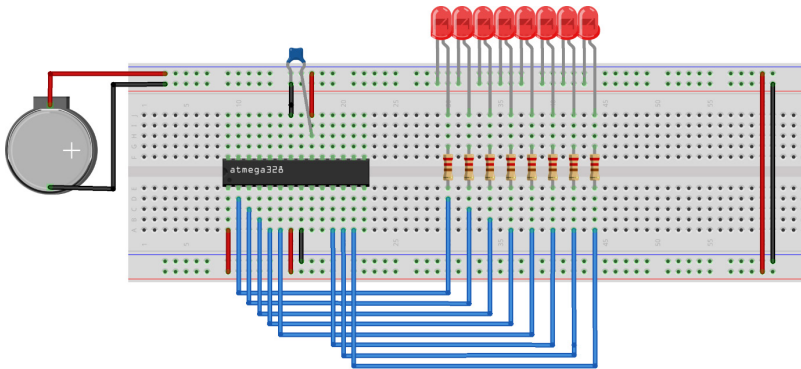
5. Adım: Devre simülasyon programında Görsel 2.73'te verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.73: 2.4 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.74'te verildiği gibi kurunuz.



Görsel 2.74: 2.4 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		

2.2.6. Mikrodenetleyiciyle Buton Kontrolü

Mikrodenetleyiciler, dış dünyadan aldıkları sinyalleri işleyerek gerçekleştirmek istedikleri işlemleri yerine getirir. Mikrodenetleyiciler, dış dünyadan sinyalleri sensörler yardımıyla alır. Sensör olarak kullanılabilen en basit / temel eleman butondur. Buton, üzerinde bulunan kontak sayesinde elektrik akımını üzerinden geçirir ya da geçirmez.

Mikrodenetleyici, dijital veri okuma işlemlerinde PINx yazmaçlarındaki değerleri kullanılır. PINx yazmaçları 8 bitlik değere sahiptir fakat bu bit değerlerine tek tek değer atama (=) operatörüyle ulaşamaz. Dijital çıkışta olduğu gibi dijital girişte de bitwise operatörleri kullanılır.

PINx yazmacından tek bir bit değerini okumak için **PINx & (1<<PORTxn);** komutu kullanılır. Örneğin B portunun 3 numaralı pinini okuyarak bir değişkene atamak için aşağıdaki komut kullanılır.

degisken = PINB & (1<<PORTB3);

PINB kaydedicisinin değeri 00111010 olsun.

(1<<PORTB3) ifadesiyle 1 (00000001) değeri sola kaydırma (<<) operatörüyle 3 bit kadar sola kaydırılır. Böylece 00001000 değeri elde edilir. PINB yazmacı, bu değerle AND (&) işlemine tabi tutulur ve çıkan sonuç değişkene atanır.

Bit	7	6	5	4	3	2	1	0	DEC
PINB	0	0	1	1	1	0	1	0	58
1<<PORTB3	0	0	0	0	1	0	0	0	8
Değişken (& işleminin sonucu)	0	0	0	0	1	0	0	0	8

Değişkenden okunan değer 0 ise PINB3 değerinin LOW olduğu, 0'dan farklı ise HIGH olduğu anlaşılır. Burada değer 0 ve 0'dan farklı olması önemlidir. Burada yapılan işlemlerde PINx kaydedicisinin değerinde herhangi bir değişiklik olmaz, sadece okuma işlemi gerçekleştirilir. Karar kontrol yapıları kullanılarak dönen değerlere göre işlemler yaptırılabilir.

AVR kütüphanesindeki **avr/sfr_defs.h** dosyasında bulunan bazı makrolar, giriş çıkış işlemlerini kolaylaştırmak için eklenmiştir.

_BV() Makrosu ile Giriş-Çıkış İşlemleri

B portunun 3. bitini HIGH yapmak için aşağıdaki komut kullanılır.

```
PORTB |= _BV(3);
```

B portunun 3. bitini LOW yapmak için aşağıdaki komutlar kullanılır.

```
PORTB &= !_BV(3);
```

```
PORTB &= ~_BV(3);
```

B portunun 3. bitini okumak için aşağıdaki komut kullanılabilir.

```
PIND & _BV(3);
```

bit_is_set() ve bit_is_clear() Makroları ile Dijital Okuma İşlemleri

Bu makrolar dijital bir pinin 1 (HIGH) mi yoksa 0 (LOW) mı olduğunu denetlemek için kullanılır.

Örneğin C portunun 2 No.lu pinini okumak için aşağıdaki komut kullanılır.

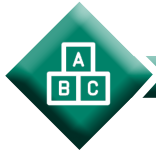
```
bit_is_set(PINC,2);
```

Bu komut işletildiğinde C portunun 2 No.lu pininde +5 V varsa 1 değerini, yok ise 0 değerini döndürür.

```
bit_is_clear(PINC,2);
```

bit_is_clear() makrosu ile okuma işlemi yapıldığında ise C portunun 2 No.lu pininde +5 V varsa 0 değerini yok ise 1 değerini döndürür.

Bu makrolarının birbirinden farkı bit_is_clear() komutunda tersleme işleminin gerçekleştirilmesidir.



UYGULAMA

Adı:	Butonla LED Kontrolü	No.: 2.5
Amaç:	Mikrodenetleyiciyle giriş-çıkış işlemleri yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, PBO pinine bağlı olan LED'i PBO pinine bağlı olan butonla yakıp söndüren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 2.12'de belirtilmiştir.

Tablo 2.12: 2.5 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı	1 adet
Direnç	220 Ω	2 adet
Direnç	10 K Ω	1 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	11 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio'da File menüsünden New Project seçeneğiyle Görsel 2.61'de görülen şekilde, GCC C Executable Project seçeneğini seçtikten sonra Name kısmına **BUTON_LED_KONTROLU** yazınız.

2. Adım: Görsel 2.62'de görülen Device Selection penceresinden **Atmega328P** mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
```

```
int main(void)
{
```

```
    DDRD=0x00;
    DDRB=0x01;
    int degisken;
```

```
// D portunun tüm pinleri giriş olarak ayarlandı.
// PBO pini çıkış olarak ayarlandı.
// PBO pinine bağlı butonun değerini tutacak
// değişkendir.
```

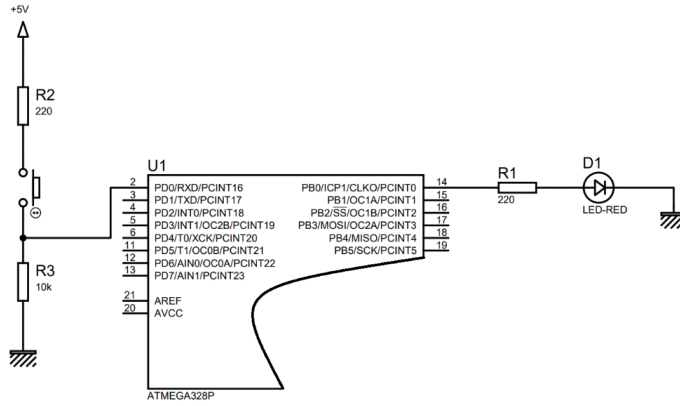
```

while (1)
{
    _delay_ms(100);
    degisken=PIND&(1<<PORTD0);    // PD0 pininde dijital okuma yapıldı ve sonuç
                                   // değişkene atandı.
    if(degisken)                   // Butona basıldı ise
    {
        PORTB=0x01;               // PD0 pini +5 V ile beslendi (LED yandı.).
    }
    else                           // Butona basılmadı ise
    {
        PORTB=0x00;               // PD0 pini 0 V ile beslendi (LED söndü.).
    }
}
}

```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz.

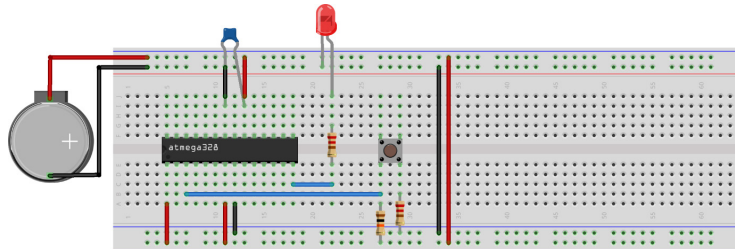
5. Adım: Devre simülasyon programında Görsel 2.75'te verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.75: 2.5 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodnetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.76'da verildiği gibi kurunuz.



Görsel 2.76: 2.4 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

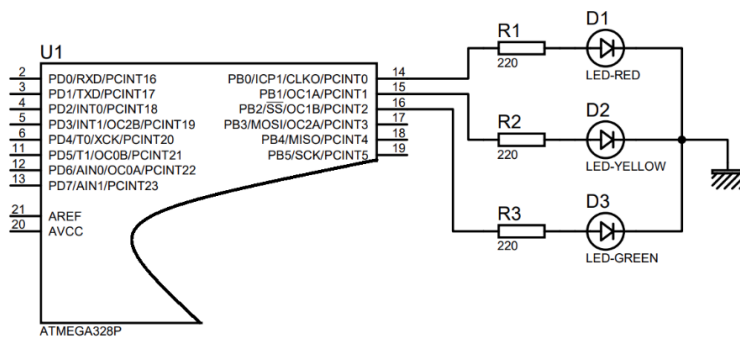
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

Görsel 2.77’de verilen devredeki mikrodenetleyicinin PB3 ve PB4 pinlerine yeşil ve sarı LED ekleyerek uygulamaya yaya ışıklarını ilave ederek devreyi oluşturunuz ve gömülü yazılımını kodlayınız.



Görsel 2.77: Uygulama devresi



UYGULAMA

Adı:	Butonla Yürüyen Işık Uygulaması	No.: 2.6
Amaç:	Mikrodenetleyiciyle giriş-çıkış işlemlerini gerçekleştirmek.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, B portuna bağlı olan sekiz adet LED’i PD0 pinine bağlı olan butonla PB0’dan PB7’ye doğru (İLERİ) yürüyen; PD1 pinine bağlı olan butonla LED’leri, PB7’den PB0’a doğru yürüyen (GERİ) devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 2.13’te belirtilmiştir.

Tablo 2.13: 2.6 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı	8 adet
Direnç	220 Ω	10 adet
Direnç	10 K Ω	2 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	20 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da File menüsünden New Project seçeneğini seçiniz. Görsel 2.61’de görülen şekilde, GCC C Executable Project seçeneğini seçtikten sonra Name kısmına **BUTONLA_YURUYEN_ISIK** yazınız.

2. Adım: Görsel 2.62’de görülen Device Selection penceresinden **Atmega328P** mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
```

```
int main(void)
{
```

```

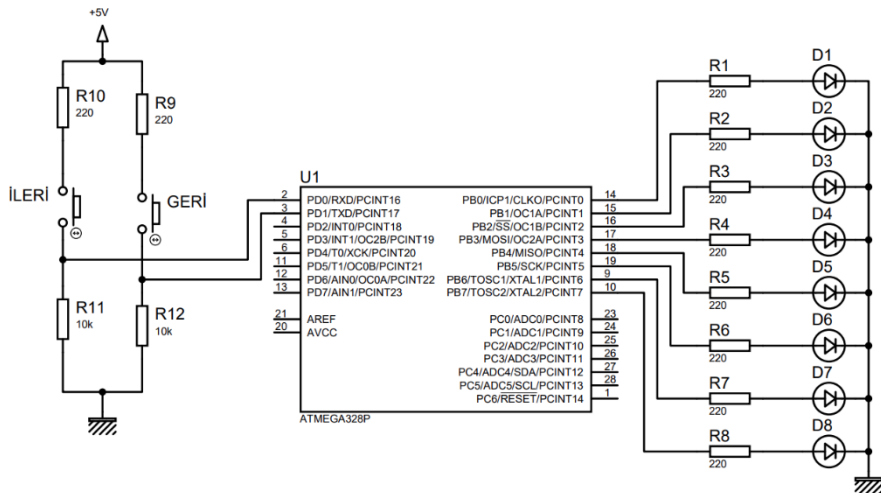
        DDRD=0x00;           // D portunun tüm pinleri giriş olarak ayarlandı.
        DDRB=0xFF;           // B portunun tüm pinleri çıkış olarak ayarlandı.
        int buton1;           // PD0 pinine bağlı butonun değerini tutacak değişken,
        int buton2;           // PD0 pinine bağlı butonun değerini tutacak değişken,
        PORTB = 0x01;         // PORTB kaydedicisinin ilk değerini ata.
        while (1)
        {
            _delay_ms(200);    // Mikrodenetleyiciler saniyede binlerce işlem
                                // yapabilen elemanlardır. Düğmeye basılıp işlem yapıldıktan sonra aynı işlemleri tekrarlamaması için
                                // _delay_ms(200= komutu ile 200 milisaniye gecikme ekliyoruz.
            buton1=PIN_D&(1<<PORTD0); // PD0 pininde dijital okuma yapıldı ve sonuç
                                        // değişkene atandı.
            buton2=PIN_D&(1<<PORTD1); // PD1 pininde dijital okuma yapıldı ve sonuç
                                        // değişkene atandı.

            if(buton1)          // PD0 pinine bağlı butona (İLERİ) basıldı ise
            {
                PORTB<<=1;      //PORTB yazmacının bitlerini bir bit sola kaydır.
            }
            if(buton2)          // PD1 pinine bağlı butona (GERİ) basıldı ise
            {
                PORTB>>=1;      //PORTB yazmacının bitlerini bir bit sağa kaydır.
            }
        }
    }
}

```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz.

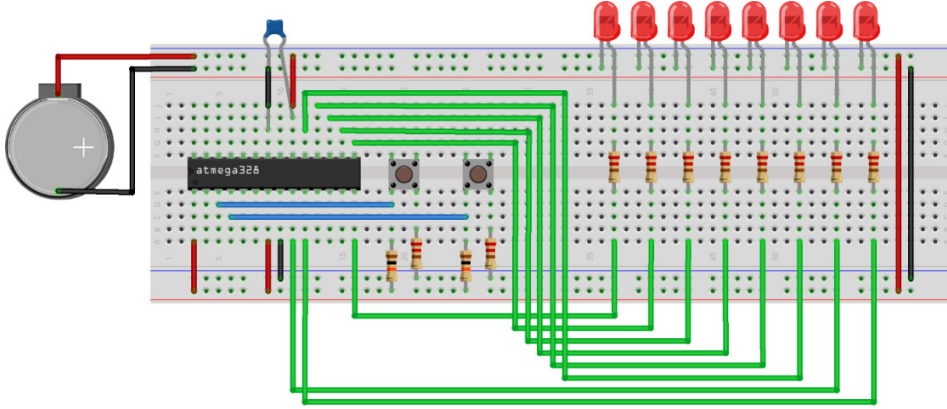
5. Adım: Devre simülasyon programında Görsel 2.78’de verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.78: 2.6 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.79’da verildiği gibi kurunuz.



Görsel 2.79: 2.6 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

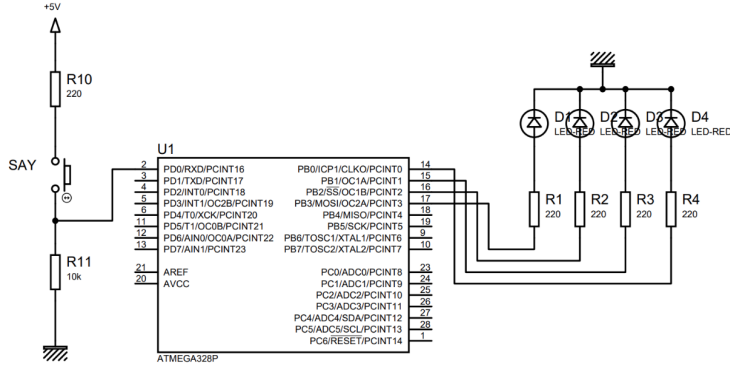
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştırıldı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

PD0 pinine bağlı olan butona her basıldığında, PB0, PB1, PB2 ve PB3 pinlerine bağlı olan LED'ler ile 0-9 arası desimal sayıların 4 bitlik binary karşılığını sırasıyla gösteren devreyi (Görsel 2.80) oluşturunuz ve gömülü yazılımını kodlayınız.



Görsel 2.80: Uygulama devresi

2.3. MİKRODENETLEYİCİYE PROGRAMI YÜKLEYEREK TEST ETME

Önceden yazılmış bir programı denetleyiciye yükleyebilmek için programlayıcıya ihtiyaç duyulur. AVR mikrodenetleyicilerine, derlenen kodu yüklemek için genellikle iki yöntem kullanılır. Bunlar, seri iletişim ve ISP yöntemleridir. Görsel 2.81'de piyasada bulunan programlayıcılar verilmiştir. Bu programlayıcılardan uygun olan seçilebilir.



Görsel 2.81: Denetleyici programlayıcıları

2.3.1 Programlayıcının Mikrodenetleyiciye Bağlantısı

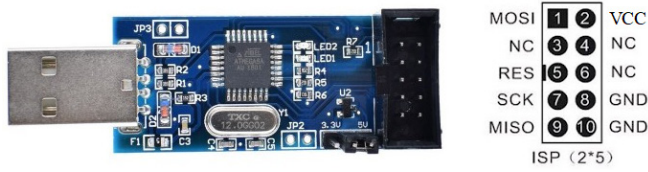
ISP, daha çok boş olarak satın alınan AVR denetleyicilerini programlamak için kullanılan yöntemdir. Bu öğrenme birimindeki uygulamalarda Görsel 2.82'de gösterilen programlayıcı kullanılacaktır. Bu programlayıcı, AVR mikrodenetleyicileri için geliştirilmiş USB ara yüzüne sahip bir ayardır.

ISP yöntemini kullanarak AVR denetleyicilerini programlamak için tasarlanmıştır. Programlayıcı, yalnızca bellek (temel mikro komutlarla çalışan sürücüler) içeren bir USB sürücüsünü kullanır ve sürücü kurulumu istemez.



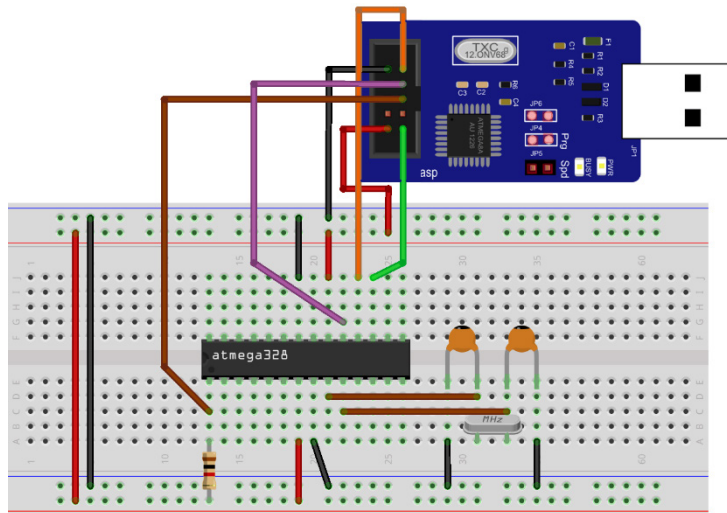
Görsel 2.82: Denetleyici programcısı

Yazılan programı denetleyiciye yüklemeyi önce uygulama devresinin kurulu olması ve programlayıcı ile denetleyicinin bağlantısının yapılmış olması gerekir. Görsel 2.83'te programlayıcının çıkış pinleri ve görevleri verilmiştir.



Görsel 2.83: Programlayıcı çıkış pinleri

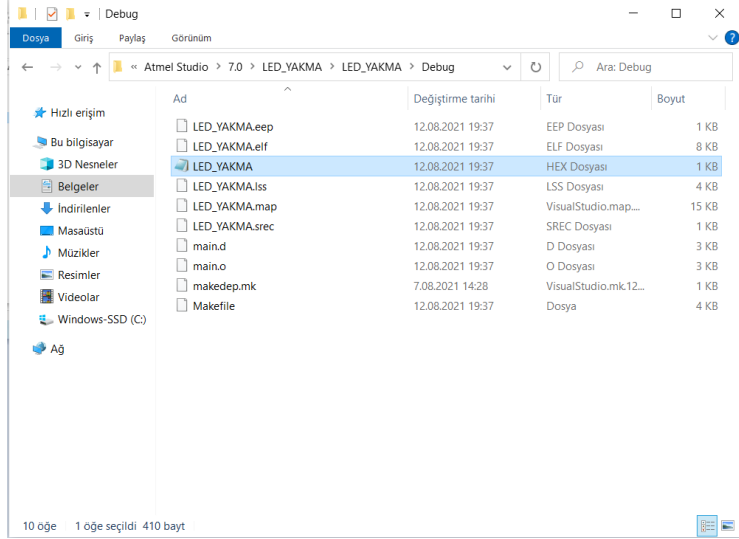
Atmega328P denetleyicisi fabrikadan haricî kristal ayarlanmış olarak gelmektedir. Programlayıcıyla Atmega328P denetleyicisini programlarken denetleyiciye haricî osilatör devresi bağlamak gerekir. Atmega328P denetleyiciyi programlamak için breadboard üzerinde Görsel 2.84'te verilen devre kurulmalıdır.



Görsel 2.84: Mikrodenetleyicinin programlayıcıya bağlantısı

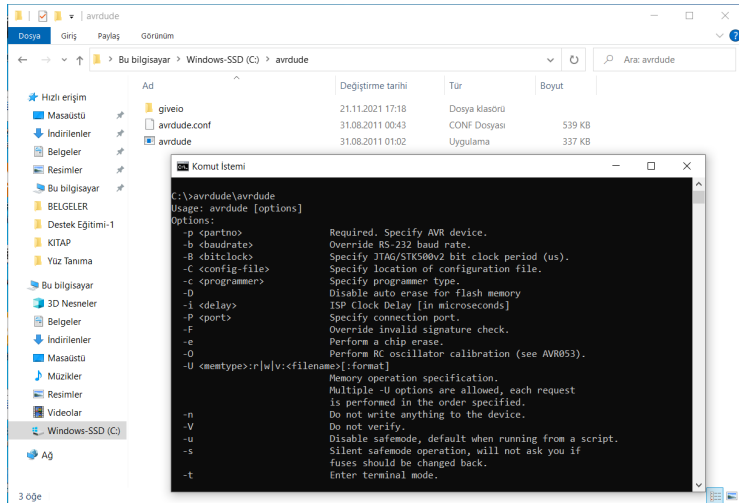
2.3.2. Mikrodenetleyiciye Programın Yüklenmesi

Yazılan programı derledikten sonra Microchip Studio programı, hex dosyası oluşturacaktır. Hex dosyası projenin kayıtlı olduğu dizinde, **Debug** klasörünün içerisinde saklanır (Görsel 2.85).



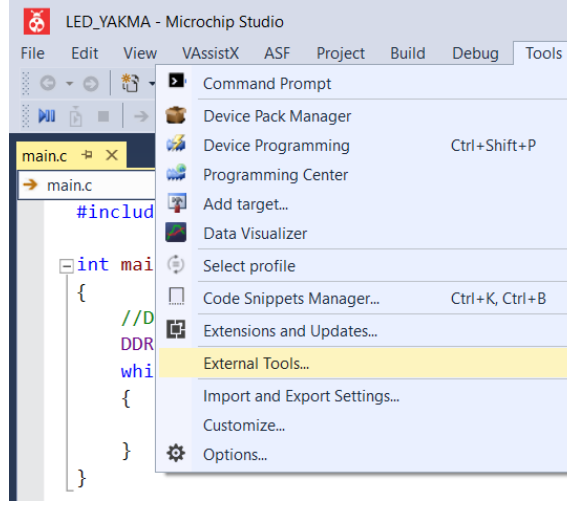
Görsel 2.85: Debug klasörü

Denetleyicinin çalışması için hex dosyasının denetleyicinin FLASH Belleğine yüklenmesi gerekir. Bunun için AVRDUDE yazılımı kullanılır. AVRDUDE yazılımı, komut satırında çalışan bir yazılımdır. <https://www.nongnu.org/avrdude/> adresinden AVRDUDE yazılımının gerekli dosyaları indirilebilir. İndirilen dosyalar C:/ dizinine kopyalanarak Görsel 2.86'da verildiği gibi komut satırında **avrdude** komutu verilerek gerekli bilgiler alınabilir.



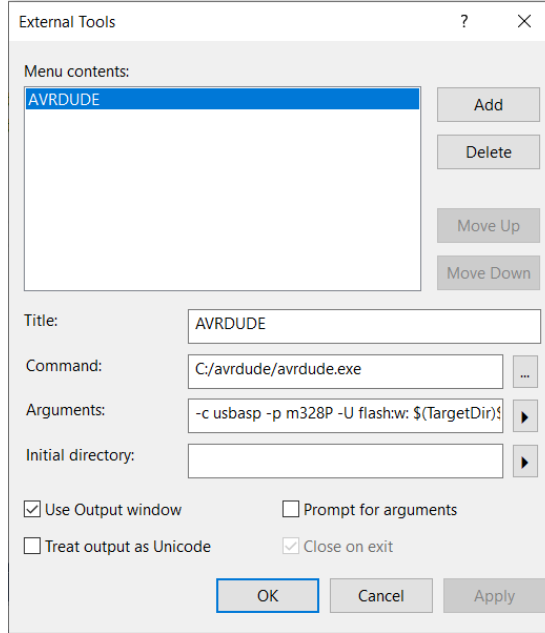
Görsel 2.86: AVRDUDE' nin komut satırında çalıştırılması

AVRDUDE yazılımı, Microchip Studio yazılımından da kullanılabilir. Bunun için **Tools** menüsünden (Görsel 2.87) **External Tools** komutu verilir.



Görsel 2.87: External Tools komutu

External Tools komutu verildikten sonra Görsel 2.88’de verilen **External Tools** penceresindeki **Add** butonuna basarak yeni araç (USBasp ve AVRDUDE), programa tanıtılır.



Görsel 2.88: External Tools penceresi

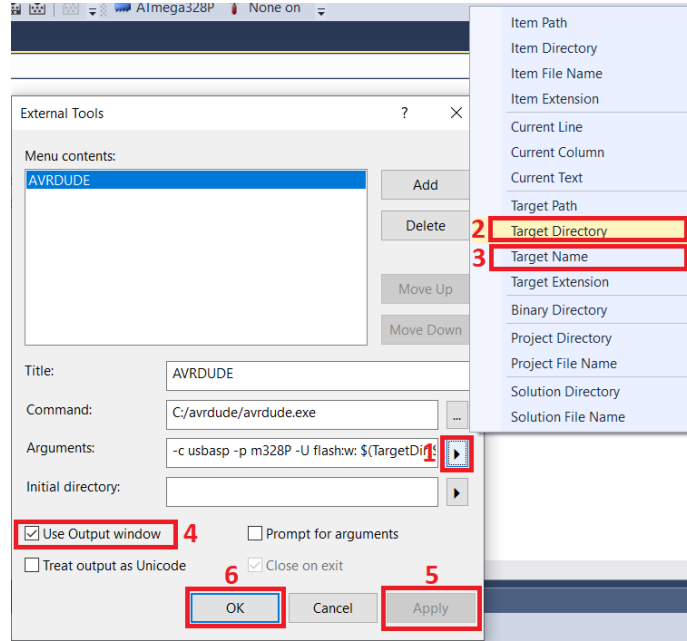
External Tools penceresindeki alanlar, aşağıdaki bilgilerle doldurulmalıdır.

Title: Araca verilecek başlık alanıdır. Herhangi bir başlık verilebilir fakat **Türkçe karakter kullanılmamalıdır**. Bu alana yazılan başlık, Microchip Studio programının araç çubuğunda ve Tools menüsünde görülecektir.

Command: Denetleyiciye yüklenecek programın konumu bu alana yazılır. AVRDUDE programının dosyaları **C:/avrdude/** dizinine kopyalandı. Command alanına **C:/avrdude/avrdude.exe** yazarak kullanılacak programın konumu belirtilir.

Arguments: Burası, avrdude komutu kullanılırken verilecek parametrelerin yazılacağı alandır.

Bu alana **-c usbasp -p m328p -U flash:w:** yazıp hemen devamında boşluk bırakılır ve Arguments alanının sonunda bulunan ok (Görsel 2.89, 1) butonuna basarak sırasıyla **Target Directory** ve **Target Name** (Görsel 2.89, 2 ve 3) seçilir.



Görsel 2.89: Argümanların seçimi

Gerekli argümanlar seçildikten sonra Arguments alanı, **-c usbasp -p m328p -U flash:w:\$(TargetDir)\$(TargetName)** olacaktır. Hemen devamına **.hex:i** yazarak argüman satırı oluşturulur.

Arguments alanı son olarak

-c usbasp -p m328p -U flash:w: \$(TargetDir)\$(TargetName).hex:i olmalıdır.

-c usbasp ile programlayıcının usbasp olduğu,

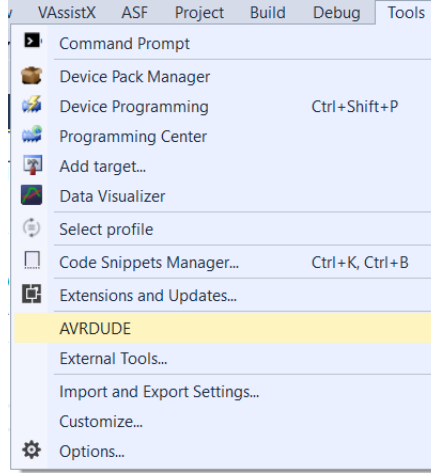
-p m328p ile programlanacak mikrodenetleyicinin Atmega328P olduğu,

-U flash:w: ile programlama işleminin denetleyicinin FLASH'ına yazmak olduğunu,

\$(TargetDir)\$(TargetName).hex:i ile hedef hex dosyasının adının ne olduğu ve hangi konumda bulunduğu Microchip Studio programına bildirilmiş olur.

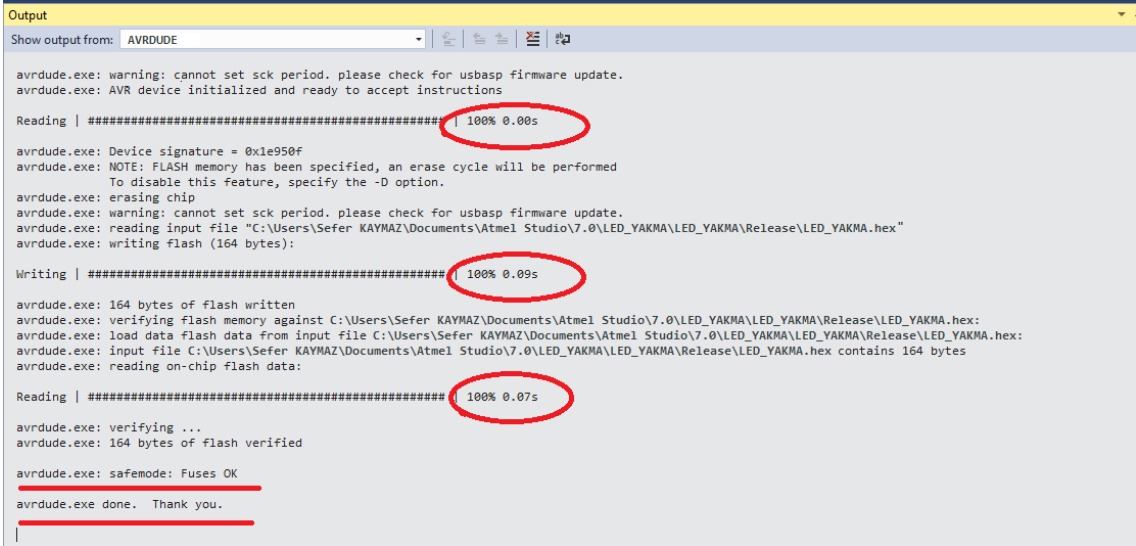
External Tools penceresindeki **Use Output window** (Görsel 2.89, 4) seçeneği işaretlenerek Microchip Studio'nun Output penceresinde avrdude yazılımından gelen mesajların görülmesi sağlanır.

Apply ve ardından **OK** butonlarına (Görsel 2.89, 5 ve 6) basılarak External Tools ekranı kapatılır. Tools menüsüne tekrar girildiğinde External Tools penceresinde Title alanına yazılan AVRDUDE başlığının bu menüde yer aldığı (Görsel 2.90) görülür.



Görsel 2.90: Tools menüsü

Görsel 2.90’da verilen Tools menüsünden, AVRDUDE seçildiğinde denetleyiciye yükleme işlemi yapılır ve yükleme sonuçları Output penceresinde görülür. Denetleyici, programlayıcıya doğru bağlandığında ve Microchip Studio programına doğru tanıtıldığında Output penceresinde Görsel 2.91’de verilene benzer bir çıktı alınır.



Görsel 2.91: Output penceresi

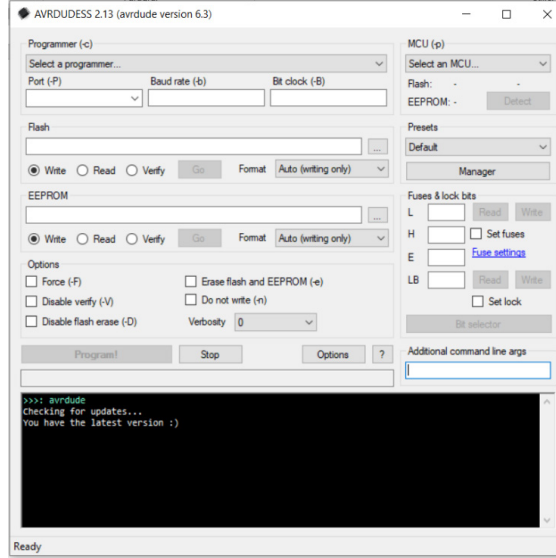


SIRA SİZDE

AVRDUDE yazılımını <https://www.nongnu.org/avrdude/> adresinden indiriniz ve Microchip Studio programına tanıtınız.

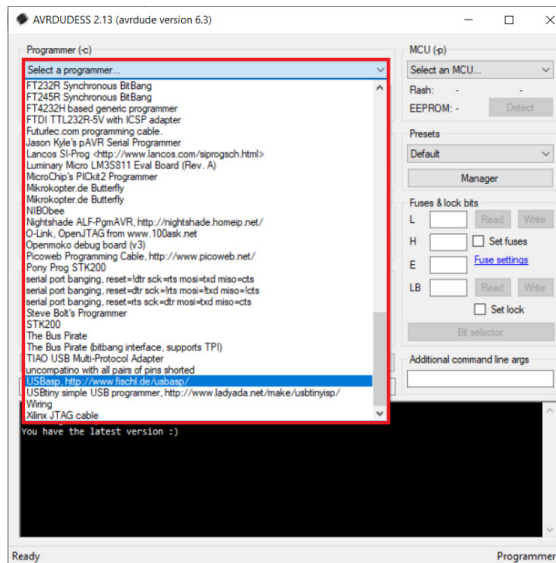
2.3.3. Mikrodenetleyicinin Osilatör ve Sigorta Ayarlarını Yapmak

AVRDUDE yazılımı kullanılarak yükleme işlemi yapılırken parametreler verilmesi gerekir çünkü AVRDUDE, komut satırında çalışan bir yazılımdır. AVRDUDESS programı ise AVRDUDE programını arka planda kullanan bir yazılımdır. Grafik arayüzüne sahip (Görsel 2.92) bu program, kullanıcılara kolaylık sağlar.



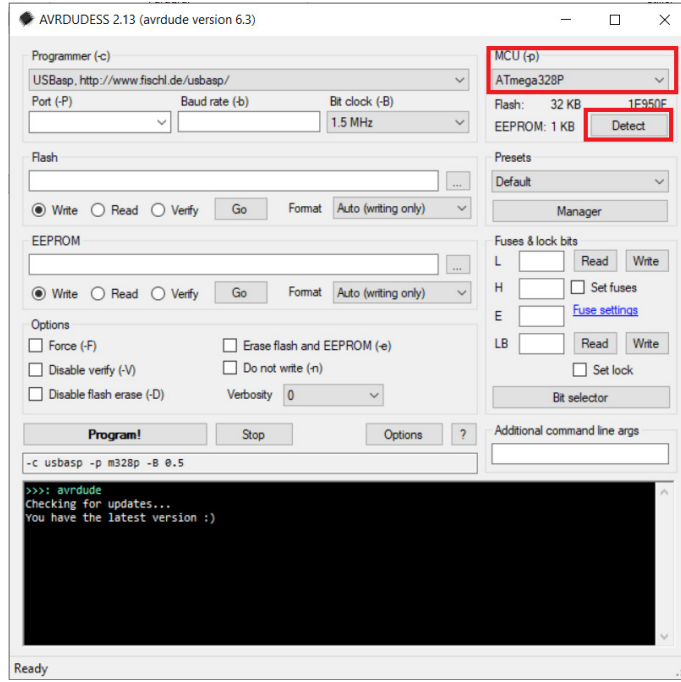
Görsel 2.92: AVRDUDESS programının arayüzü

Görsel 2.92’de verilen pencerede yer alan **Fuses & lock bits** alanında gerekli parametreler verilerek osilatör ve sigorta ayarları yapılabilmektedir. AVRDUDESS yazılımı kullanılarak denetleyiciye program yükleme esnasında sigorta ve osilatör ayarlaması yapılabilir. Öncelikle programlayıcı cihaz seçilmelidir. Bunun için Görsel 2.93’te verildiği gibi **Programmer(-c)** açılır listesinden USBasp cihazı seçilir.



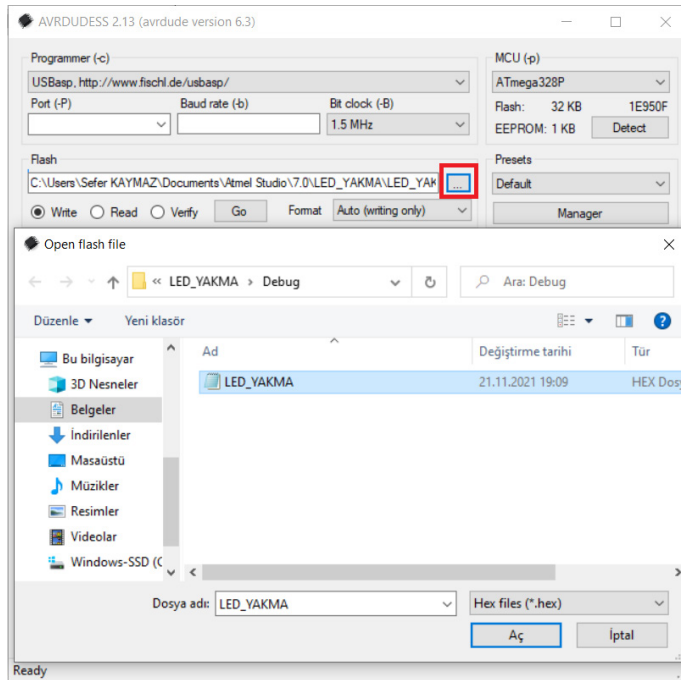
Görsel 2.93: Programlayıcı cihazının seçimi

MCU(-p) açılır listesinden Atmega328P denetleyicisi seçilir ya da **Detect** tuşuna basarak (Görsel 2.94) programın, bilgisayarın portlarını tarayarak denetleyiciyi otomatik bulması sağlanabilir.



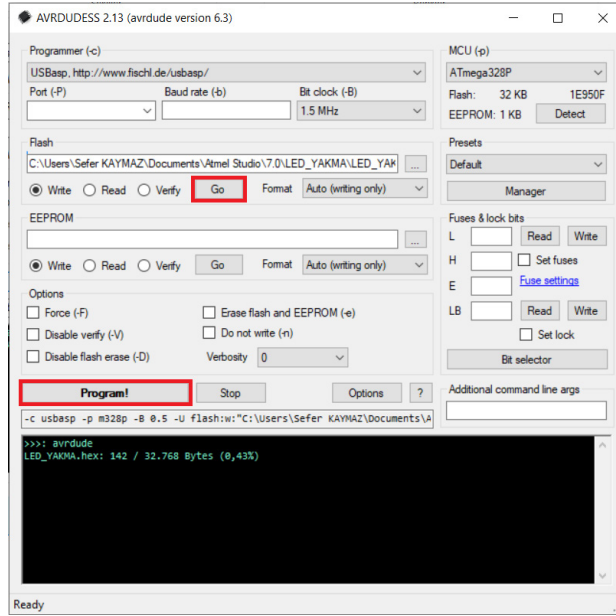
Görsel 2.94: Uygun mikrodenetleyicinin seçimi

Flash belleğe gönderilecek hex dosyası Görsel 2.95'te işaretlenen butona basılarak seçilir.



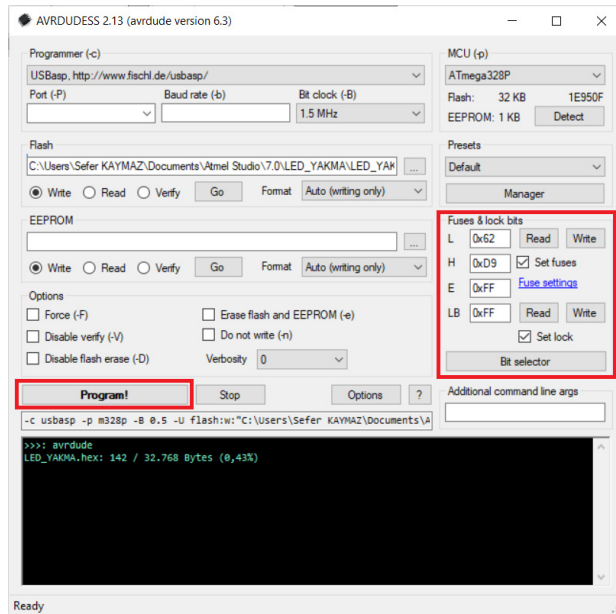
Görsel 2.95: Hex dosyasının seçimi

Görsel 2.96’da verilen pencerede işaretli olan butonlar denetleyiciye hex dosyasını yazmak için kullanılan butonlardır.



Görsel 2.96: Programı denetleyiciye yükleme

Yüklenecek dosya seçildikten sonra Flash alanındaki **Write** seçiliyken **Go** butonuna basılırsa Flash’a yazma işlemi gerçekleştirilir fakat osilatör ve sigortalarda bir ayarlama yapılmaz. Yüklenecek hex dosyası tekrar seçildikten sonra **Fuses & lock bits** alanından **Set fuses** ve **Set lock** seçenekleri seçili değilken **Program!** butonuna basılırsa Flash’a yazma işlemi gerçekleştirilir fakat osilatör ve sigortalarda yine bir ayarlama yapılmaz.



Görsel 2.97: Fuses ve lock bitleri

Program! butonuna basmadan önce **fuses** ve **lock** bitlerinin doğru ayarlanmış olduğuna emin olunmalıdır. Bu bitler, programlama esnasında yanlış ayarlanırsa denetleyici kullanılmaz hâle gelebilir. Görsel 2.97'deki gibi **Set fuses** ve **Set lock** seçili ise bu bitler, belirtilen değerlere göre ayarlanacaktır.

AVR mimarisine sahip denetleyicilerin sigorta ayarlamaları, kodlama yapılan yazılımda, kod satırlarıyla düzenlenemez. PIC denetleyicilerde kodlama esnasında konfigürasyon ayarlaması yapılabilirken AVR denetleyicilerinde, konfigürasyon bitlerini, programı yüklediğimiz yazılımlarla ayarlayabiliriz.

AVR mikrodenetleyicilerinin sigorta düzenlemelerini gerçekleştirirken yapılan değişikliklerden emin olunamamışsa değişiklikler uygulanmamalıdır. Aksi takdirde denetleyici bir daha programlanamaz hâle gelebilir. Dikkat edilmesi gereken bir diğer husus da 1 ve 0'ın sigorta konfigürasyonundaki anlamıdır. 1, **unprogrammed (ayarlanmamış)** anlamına gelirken 0, **programmed (ayarlanmış)** anlamına gelir.

Atmega328P denetleyicisinde üç baytlık ($3 \times 8 = 24$ bit) sigorta bilgisi vardır. Bunlar:

- Low byte fuse
- High byte fuse
- Extended byte fuse

2.3.3.1 Low Byte Fuse

Low byte fuse bitleri, denetleyicinin saat kaynağıyla (osilatör) ilgili ayarlamalarının yapıldığı ayar sigortalarıdır.

Görsel 2.98'de low byte fuse içeriği verilmiştir.

Fuse	7	6	5	4	3	2	1	0	Bit
Low	CKDIV8	CKOUT	SUT1	SUT0	CKSEL3	CKSEL2	CKSEL1	CKSEL0	Name
Byte	0	1	1	0	0	0	1	0	Default

Görsel 2.98: Low byte fuse

Varsayılan değer olarak 01100010 = 0x62 değeri kullanılır.

CKDIV8: Denetleyicinin aktif osilatör hızının sekize bölünüp kullanılacağını belirleyen sigortadır. CKDIV8 programlanmış (0) ise seçili olan osilatör frekansı sekize bölünerek kullanılır. Örneğin osilatör kaynağı olarak kalibre edilmiş dâhilî 8 MHz RC osilatörü kullanılıyor ve CKDIV8 biti programlanmış (0) ise denetleyicinin çalışma frekansı $8 \text{ MHz} / 8 = 1 \text{ MHz}$ olur.

Kullanılacak osilatör kaynağının frekansı, denetleyicinin çalışabileceği maksimum frekanstan daha fazlaysa CKDIV8 biti programlanarak (0) osilatör frekansı sekize bölünür. Bu sayede denetleyicinin çalışabileceği bir frekans değeri elde edilebilir.

CKDIV8 biti fabrika çıkışı varsayılan değeri 0 (programlanmış) olarak gelir.

CKOUT: Denetleyiciyle eş zamanlı çalışacak bir devre kullanılacaksa CKOUT bitini programlayarak (0) PORTB0 pininden saat sinyali alınır ve haricî devre bu sinyalle sürülebilir.

CKOUT biti varsayılan olarak programlanmamış (1) olarak gelir.

CKSEL[3..0]: Denetleyici için saat kaynağının (osilatör) seçildiği bitlerdir. Tablo 2.14'te CKSEL bitlerinin durumuna göre hangi osilatör kaynağının kullanılacağı verilmiştir.

Tablo 2.14: Osilatör Seçimi

Osilatör Kaynağı	CKSEL 3...0
Low Power Crystal Oscillator	1111 – 1000
Full Swing Crystal Oscillator	0111 – 0100
Low Frequency Crystal Oscillator	0101 – 0100
Internal 128 kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

CKSEL bitleri varsayılan olarak 0010 olarak ayarlanmıştır. Bu değer kalibre edilmiş dâhilî RC osilatörün kullanılacağı anlamına gelir. Düşük güçlü bir kristal osilatör kullanılacaksa CKSEL bitleri 1111-1000 arası ayarlanmalıdır yani 1111, 1110, 1101, 1100, 1011, 1010, 1001 veya 1000 olarak ayarlanmalıdır. Seçeneklerden hangisinin kullanılacağını ise osilatörün frekansı belirleyecektir. Tablo 2.15'te düşük güçlü kristal osilatörün frekansına göre CKSEL bitlerinin nasıl ayarlanması gerektiği verilmiştir.

Tablo 2.15: Düşük Güçlü Kristal Osilatör Frekanslarına Göre CKSEL Ayarları

Osilatör Frekansı (MHz)	CKSEL 3...1
0.4 – 0.9	100
0.9 – 3.0	101
3.0 – 8.0	110
8.0 – 16.0	111

Örneğin 8 ile 16 MHz'lik bir kristal osilatör kullanılacaksa CKSEL[3...1] bitleri 111 olarak ayarlanmalıdır. Bu durumda CKSEL0 biti ise 1 ya da 0 olabilir. 0 mı 1 mi olacağını ise SUT[0..1] bitleri belirleyecektir.

SUT[1:0]: Denetleyici besleme geriliminin istenilen seviyeye ulaşabilmesi için belli bir zamana ihtiyacı vardır. Bu zaman, osilatör kaynağına göre belirlenir. Başlatma süresinin kontrolü SUT[1..0] bitleriyle sağlanır.

Kalibre edilmiş dâhilî RC osilatörüne göre ayar yapılırken üç adet güç koşuluyla karşılaşılır. Tablo 2.16’da bu koşullar verilmiştir.

Tablo 2.16: Kalibre Edilmiş Dâhilî RC Osilatörün Güç Koşullarına Göre SUT Seçenekleri

Güç Koşulu	Kapatma – Güç Tasarrufu	Resetlemede Ek Zaman	SUT[1:0]
BOD enabled	6 CK	14 CK	00
Fast rising power	6 CK	14 CK + 4.1 ms	01
Slowly rising power	6 CK	14 CK + 65 ms	10
Reserved			11

SUT bitleri varsayılan olarak 10’a ayarlıdır. Bu durum yavaş yükselen güç içindir. Güç kapatıldıktan veya güç tasarrufu yapıldıktan sonra 6 saat çevrimi gecikme oluşur. Ayrıca resetleme işleminden sonra 14 saat çevrimine ek olarak 65 milisaniye gecikme oluşturur ve denetleyici kodları bu gecikmelerden sonra işlemeye başlar.

Tablo 2.17’de kullanılacak düşük güçlü haricî osilatör kaynağının güç koşullarına göre SUT[1:0] ve CKSELO bitlerinin alması gereken değerler verilmiştir.

Tablo 2.17: Düşük Güçlü Haricî Osilatörün Güç Koşullarına Göre SUT-CKSEL Seçenekleri

Osilatör Kaynağı - Güç Koşulu	Kapatma – Güç Tasarrufu	Resetlemede Ek Zaman	CKSELO	SUT[1:0]
Ceramic resonator, fast rising power	258 CK	14 CK + 4.1 ms	0	00
Ceramic resonator, slowly rising power	258 CK	14 CK + 65 ms	0	10
Ceramic resonator, BOD enabled	1K CK	14 CK	0	10
Ceramic resonator, fast rising power	1K CK	14 CK + 4.1 ms	0	11
Ceramic resonator, slowly rising power	1K CK	14 CK + 65 ms	1	00
Crystal Oscillator, BOD enabled	16K CK	14 CK	1	01
Crystal Oscillator, fast rising power	16K CK	14 CK + 4.1 ms	1	10
Crystal Oscillator, slowly rising power	16K CK	14 CK + 65 ms	1	11

Tablo 2.17’ye göre düşük güçlü kristal osilatör kullanılarak denetleyiciyi güvenli olarak başlatabilmek için en az 16 000 saat çevrimine ihtiyaç vardır. Bu durumda CKSELO biti 1 olarak ayarlanmalıdır. Resetleme işleminde, 14 saat çevrimine artı 65 milisaniyelik ek gecikme oluşturmak için SUT[1:0] bitleri 11 olarak ayarlanmalıdır. Özetle denetleyici, dâhilî kalibre edilmiş 8 MHz RC osilatörüyle çalıştırılmak isteniyorsa low fuse byte’ı 01100010 = 0x62 olarak ayarlanmalıdır.

Denetleyici, haricî 8-16 MHz arası düşük güçlü kristal osilatörle çalıştırmak isteniyorsa low fuse byte’ı 11111111 = 0xFF olarak ayarlanmalıdır.

2.3.3.2 High Byte Fuse

Görsel 2.99’da high byte fuse içeriği verilmiştir.

Fuse	7	6	5	4	3	2	1	0	Bit
High	RSTDISBL	DWEN	SPIEN	WDTON	EESAVE	BOOTSZ1	BOOTSZ0	BOOTRST	Name
Byte	1	1	0	1	1	0	0	1	Default

Görsel 2.99: High byte fuse

BOOTRST: Mikrodenetleyiciye bootloader programı yüklendiğinde denetleyicinin bootloader adresine yönlendirilmesi gerekir. BOOTRST biti programlandığında (0) denetleyici, reset sırasında bootlader adresine gidecektir. Varsayılan değeri programlanmamıştır (1).

BOOTSZ[1:0]: Mikrodenetleyicinin program hafızasında kaç byte, bootloader alanı ayrılacağı belirlendiği bitlerdir. Tablo 2.18’de BOOTSZ bitlerinin değerlerine göre kaç byte bootloader alanı ayrıldığı verilmiştir.

Tablo 2.18: BOOTSZ Bitlerine Göre Bootloader Boyutu

BOOTSZ1	BOOTSZ0	Boot Size
1	1	256 words
1	0	512 words
0	1	1024 words
0	0	2048 words

Varsayılan olarak BOOTSZ1 = 0, BOOTSZ0 = 1’dir yani 1 024 words alan ayrılmıştır. $1\ 024 \times 2 = 2\ 048$ byte = 2 Kbyte’tr.

EESAVE: AVR denetleyicilerine program yüklemesi yapılırken program hafızası silinir. Sonra yeni programın yükleme işlemi gerçekleştirilir. Program hafızasının silinmesi esnasında EEPROM da silinir. EESAVE biti programlanırsa (0) program yüklenmesi esnasında EEPROM’da bulunan veriler silinmez ve bu sayede önceden kaydedilmiş veriler saklanmış olur. EESAVE bitinin varsayılan değeri programlanmamıştır (1).

EEPROM’un belli bir yazma / silme ömrü vardır. EEPROM ile işlem yoksa EESAVE biti programlanmalıdır (0). EEPROM’un ömrü boşa kullanılmamış olur.

WDTON: Watchdog’u aktif veya pasif yapmak için kullanılan sigorta bitidir. WDTON bitinin varsayılan değeri programlanmamıştır (1). Watchdog yazılımla da aktif edilebilir.

SPIEN: Atmega mikrodenetleyicileri daha çok seri programlayıcılarla programlanmaktadır. Seri programlayıcılar SPI protokolünü kullanır. SPIEN biti programlanmış (0) ise denetleyici, seri programlayıcılarla programlanabilir. SPIEN biti programlanmamış (1) ise denetleyici, seri programlayıcılarla programlanamaz. Tekrar programlamak için yüksek gerilimli paralel / seri programlayıcılar kullanmak gerekir. SPIEN bitinin varsayılan değeri SPIEN = 0’dır.

DWEN: Debug işlemi normalde kodlama yazılımı üzerinden gerçekleştirilir. Atmel mikrodenetleyicilerinde chip-debugging (OCD) özelliği mevcuttur. Yüklenen program chip üzerinde Debug edilebilir. DWEN biti programlanmış (0) yapılırsa denetleyicinin reset pini, Debug için haberleşme pini olarak kullanılır fakat reset pini, SPI programlayıcılar için gerekli bir pindir. Reset pini kullanılmazsa SPI programlayıcılarla denetleyici programlanamaz. DWEN bitinin varsayılan değeri DWEN = 1'dir.

RSTDISBL: Reset pini, giriş / çıkış (I/O) pini olarak da kullanılabilir. RSTDISBL biti programlanırsa (0) reset pini, normal I/O pini olarak kullanılır. Unutulmaması gereken SPI programlayıcıların reset pinine olan gereksinimidir. Reset pini devre dışı kalırsa denetleyici, bir daha SPI programlayıcılarıyla programlanamaz. RSTDISBL bitinin varsayılan değeri RSTDISBL = 1'dir.

Genel olarak high byte fuse değeri varsayılan olarak kullanılmalıdır. High byte fuse değeri varsayılan olarak 11011001 = 0xD9'dur.

2.3.3.3 Extended Byte Fuse

Extended byte fuse ayarları sadece BOD (Brown-Out Detection) seviyesini programlamak için kullanılır. Görsel 2.100'de extended byte fuse içeriği verilmiştir.

Fuse	7	6	5	4	3	2	1	0	Bit
Extended	-	-	-	-	-	BODLEVEL2	BODLEVEL1	BODLEVEL0	Name
Byte	1	1	1	1	1	1	1	1	Default

Görsel 2.100: Extended byte fuse

Atmega328P denetleyicisi belirli gerilim değerlerinde stabil olarak çalışır. Bu gerilim değerleri sağlanmazsa denetleyici, istenilen şekilde çalışmayacaktır. BOD aktif edildiğinde bu gerilim değerleri kontrol edilir. İstenilen gerilim değerleri sağlanmazsa denetleyici resetlenerek olumsuz durumların oluşması engellenmiş olur.

Tablo 2.19'da BOODLEVEL bitlerinin durumuna göre BOD seviyeleri verilmiştir.

Tablo 2.19: BOODLEVEL-BOD Seviyeleri

BOODLEVEL[2..0]	Min VBOT	Typ VBOT	Max VBOT	Birim
111	BOD Disabled			
110	1.7	1.8	2.0	Volt
101	2.5	2.7	2.9	
100	4.1	4.3	4.5	
011	Reserved			
011				
001				
000				

BOODLEVEL bitlerinin varsayılan değerleri 111'dir yani BOD devre dışı bırakılmıştır.

Özet olarak Atmega 328P mikrodenetleyicisinin varsayılan sigorta ayarları;

Low fuse = B01100010 = 0x62,

High fuse = B11011001 = 0xD9,

Extended fuse = B11111111 = 0xFF'dir.



UYGULAMA

Adı:	Butonla BCD Sayıcı	No.: 2.7
Amaç:	Programlayıcıyla mikrodenetleyiciye kod yüklemek.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, PBO pinine bağlı olan LED'i PD0 pinine bağlı olan butonla yakıp söndüren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 2.20'de belirtilmiştir.

Tablo 2.20: 2.7 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LED	5 mm, kırmızı	1 adet
Direnç	220 Ω	2 adet
Direnç	10 K Ω	1 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	11 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio'da File menüsünden New Project seçeneğiyle Görsel 2.61'de görülen şekilde, GCC C Executable Project seçeneğini seçtikten sonra Name kısmına **BUTON_BCD_UYGULAMASI** yazınız.

2. Adım: Görsel 2.62'de görülen Device Selection penceresinden **Atmega328P** mikrodenetleyiciyi seçiniz.

```

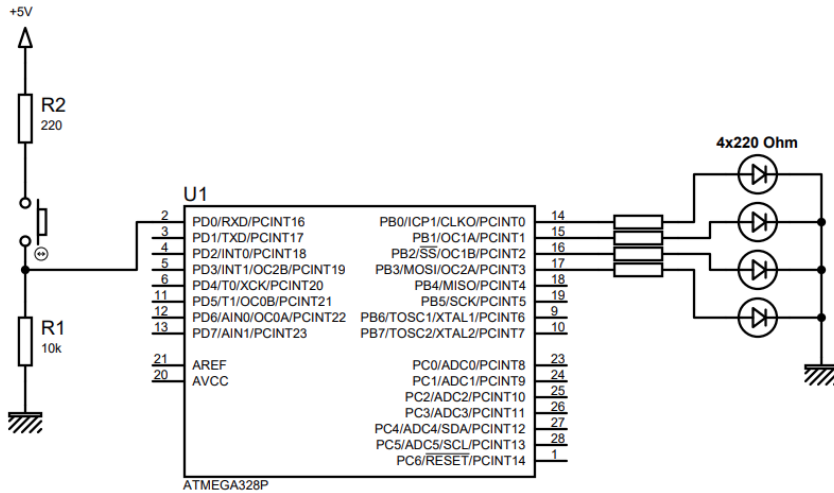
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRD=0x00;           // PORTD giriş olarak ayarlandı.
    DDRB=0xFF;           // PORTB çıkış olarak ayarlandı.
    int sayac=0;          // 0-15 arası tam sayıları tutacak değişken
    PORTB = sayac;        // B Portunun değerini sıfırla
    while (1)
    {
        _delay_ms(200);   //200 milisaniyelik gecikme oluşturuldu
        if(PIND&(1<<PORTD0)) //Butona basıldı mı?
        {
            sayac++;       //Butona basıldı ise sayaç değişkenini 1 arttır
        }
        if(sayac>15)       //sayac değişkeninin değeri 15'in üstüne çıkarsa
            sayac=0;       //sıfırla
        PORTB=sayac;       //sayac değişkeninin değerini B portuna yaz
    }
}

```

4. Adım: Yazdığınız programı, Build menüsünden Build Solution seçeneğiyle derleyiniz.

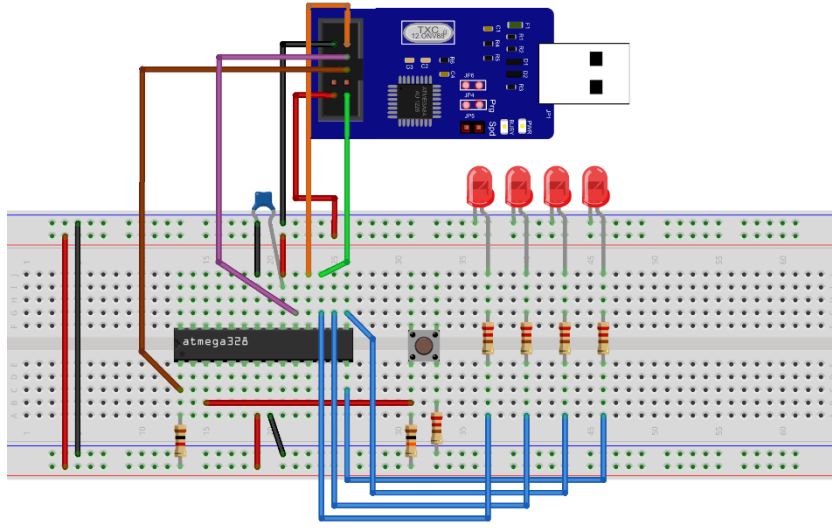
5. Adım: Devre simülasyon programında Görsel 2.101'de verilen devreyi kurunuz ve simüle ediniz.



Görsel 2.101: 2.7 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 2.102’de verildiği gibi kurunuz.



Görsel 2.102: 2.7 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımıyla denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



ÖLÇME VE DEĞERLENDİRME 2

A) Aşağıda verilen cümlelerin başındaki boşluğa, cümlede yer alan ifade doğru ise “D”, yanlış ise “Y” yazınız.

1. () Mikrodenetleyicinin yapması istenen işlemlerin kodlandığı, program kodlarının saklandığı, programlanabilir belleklere EEPROM adı verilir.
2. () Von Neumann mimarisinde veri ve program kodları, aynı bellek bloku içinde yer alır.
3. () Mikrodenetleyicinin, dış çevreyle iletişimini sağlayan pinlerinin giriş ya da çıkış olarak ayarlanmasını DDRx kaydedicileri sağlar.
4. () Mikrodenetleyiciler program belleğinde bulunan kodları yürütebilmek için kare dalga saat sinyaline ihtiyaç duyar.
5. () D portunun tüm pinlerini çıkış olarak ayarlamak için DDRD kaydedicisine 0x00 değerini yazmak gerekir.
6. () Kesme; mikrodenetleyicinin işlemekte olduğu ana programı, başka bir kaynaktan gelen sinyal sebebiyle durdurması, kesme fonksiyonuna gitmesi ve bu fonksiyon da işletilmesi gereken komutları işletmesi işlemidir.
7. () AVR mimarili mikrodenetleyicilerde sigorta ayarlaması kodlama esnasında gerçekleştirilir.
8. () Sigorta konfigürasyonunda 1 olarak programlanan sigorta, ayarlanmış (programmed) yani aktif edilmiş anlamına gelir.
9. () Program yüklemesi esnasında denetleyicinin EEPROM’unda bulunan verilerin silinmesini istemiyorsa EESAVE sigorta biti, 0 yapılmalıdır.
10. () Son yapılan aritmetik işlemin sonucunda oluşan bilgilerin kaydedildiği bellek alanı, SREG kaydedicisidir.

B) Aşağıdaki cümlelerde bulunan boşlukları uygun kelimelerle doldurunuz.

11. Veri (RAM Bellek) ve komutların (ROM Bellek) bulunduğu bellek bloklarının birbirinden ayrı olduğu mimariye mimarisi denir.
12. Zamanlamanın önemli olduğu devrelerde osilatörler tercih edilir.

13. Giriş olarak ayarlanan pinlerin +5 V ile mi 0 V ile mi beslendiği verisini tutan kaydedicisidir.
14. Mikrodenetleyicinin besleme uçlarına gerilim uygulandığında uygulanan gerilim seviyesinin istenilen düzeye çıkıncaya kadar denetleyiciyi reset konumunda bekletme, devresinin görevidir.
15. Dijital giriş / çıkış portu olarak kullanılabileceği gibi analog giriş için de kullanılabilen port, portudur.
16. Dijital sinyali lojik 1 ve lojik 0 seviyesinde, belirli zaman aralıklarında tutarak analog sinyal elde edilmesini sağlayan tekniğe adı verilir.
17. Çıkış (Output) olarak ayarlanan pinin 5 V (lojik 1, HIGH) ile mi 0 V (lojik 0, LOW) ile mi besleneceği verisini tutan kaydedicisidir.
18. Gecikme fonksiyonlarını kullanabilmek için kütüphanesinin projeye dâhil edilmesi gerekir.
19. Geçici verilerin saklandığı, denetleyicinin enerjisi kesildiğinde içerisindeki verilerin silindiği bellek alanı
20. Denetleyicinin PORTB0 pini, denetleyici ile eş zamanlı çalışacak devre için saat sinyali çıkışı olarak kullanmak istendiğinde biti 0 yapılmalıdır.

C) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

21. Mikrodenetleyiciler, kare dalga saat sinyalini üretebilmek için aşağıdaki devrelerden hangisini kullanır?

- A) ADC B) DCA C) Osilatör D) Reset E) Zamanlama

22. Mikrodenetleyiciyi hangi reset kaynağının resetlediği hakkında bilgileri tutan kaydedici, aşağıdakilerden hangisidir?

- A) DDRx B) MCUSR C) PINx D) PORTx E) SREG

23. Bilişim Teknolojileri alan öğrencisi Ali, mikrodenetleyicinin PD3 pinine bağlı olan LED'i yakmak istemektedir.

Ali, denetleyicinin PD3 pinine enerji göndermek için aşağıda verilen kodlardan hangisini kullanmalıdır?

- A) DDRD = 0b00000100; B) PIND = 0b00000100;
 C) PORTD = 0b00000100; D) PORTD = 0b00100000;
 E) PORTD = 0b11111011;

24. Yazılan kodun, makine diline dönüştürülmesi için derlenmesi gerekmektedir.

Kod derlemek için aşağıda verilen menülerden hangisi kullanılır?

- A) Build B) Edit C) File D) Project E) Tools

25. D portunun düşük değerli 4 pinini çıkış, yüksek değerlikli 4 pinini giriş olarak ayarlamak isteyen bir programcı, aşağıda verilen tanımlamalardan hangisini yapmalıdır?

- A) DDRD = 0x0F; B) DDRD = 0xF0;
C) PIND = 0x0F; D) PIND = 0xF0
E) PORTD = 0xF0;

26. İki kod arasında 1 saniye gecikme oluşturmak isteyen bir yazılımcı aşağıda verilen kodlardan hangisini kullanmalıdır?

- A) _delay_ms(1); B) _delay_ms(1000);
C) _delay_us(1); D) _delay_us(100);
E) _delay_us(1000);

27. DDRC = 0b00101100; kodunu, heksadesimal formata çevirmek isteyen bir programcının yazacağı kod aşağıdakilerden hangisidir?

- A) DDRD = 0x12 B) DDRD = 0x2C
C) DDRD = 0h2C D) DDRD = 0x3B
E) DDRD = 0xC2

28. Aşağıdakilerden hangisi Atmega328P mikrodenetleyicisinin saat kaynağı ile ilgili ayarlamaların yapıldığı sigorta bayttır?

- A) Extend B) High C) Lock D) Low E) PORTx

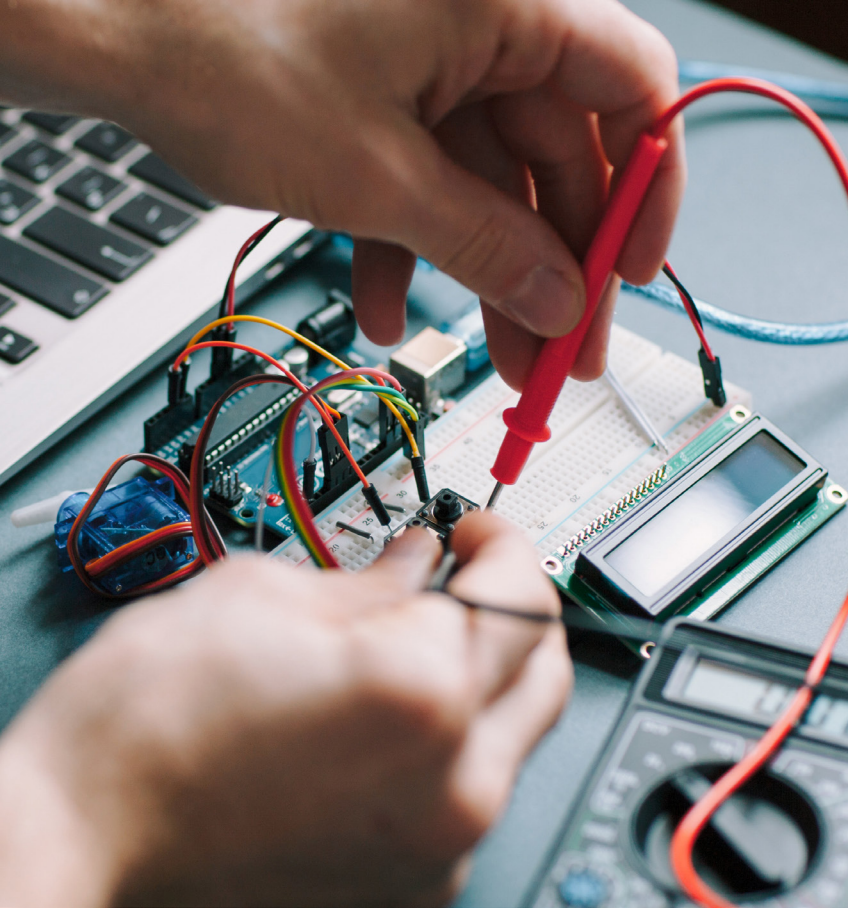
29. Aşağıda verilen "High Byte Fuse" bitlerinden hangisi, reset esnasında mikrodenetleyiciyi program belleğinde bulunan bootloader yazılımının adresine yönlendiren sigorta bitidir?

- A) BOTRST B) BOOTSZ1
C) EESAVE D) RSTDISBL
E) SPIEN

30. Aşağıda verilen "High Byte Fuse" bitlerinden hangisi, mikrodenetleyicinin reset pinini normal giriş / çıkış pini olarak yönlendirmeyi sağlayan sigorta bitidir?

- A) BOTRST B) BOOTSZ1
C) EESAVE D) RSTDISBL
E) SPIEN

3. ÖĞRENME BİRİMİ



**MİKRODENETLEYİCİ
İLE ÇEVRE BİRİMLERİ
BAĞLAMA**



KONULAR

- 3.1. MİKRODENETLEYİCİ İLE TUŞ TAKIMINDAN VERİ OKUMA
- 3.2. MİKRODENETLEYİCİ İLE DISPLAY KONTROLÜ
- 3.3. MİKRO DENETLEYİCİ İLE RÖLE KONTROL UYGULAMALARI
- 3.4. MİKRODENETLEYİCİ İLE MOTOR KONTROLÜ
- 3.5. MİKRODENETLEYİCİ İLE HABERLEŞME UYGULAMALARI

NELER ÖĞRENECEKSİNİZ?

- Mikrodenetleyiciye uygun tuş takımı seçimi
- Tuş takımına göre program yazılımı
- Mikrodenetleyiciye uygun display seçimi
- Displaye göre program yazılımı
- Mikrodenetleyiciye uygun röle seçimi
- Röle kontrol programı yazılımı
- Mikrodenetleyiciye uygun motor seçimi
- Motor kontrol programı yazılımı
- Mikrodenetleyiciye uygun haberleşme çeşitleri
- Haberleşme programı yazılımı

ANAHTAR KELİMELER

DC motor, display, haberleşme, H-Köprüsü, Karakter LCD, keypad, master, Matris LED, Mikro işlemci, mikrodenetleyici, motor, röle, slave, SPI, step motor, tuş takımı, TWI, USART, 7 Segment Display.

HAZIRLIK ÇALIŞMALARI

1. Basit bir tuş takımında, hangi tuşa basıldığı sistem tarafından nasıl anlaşılır?
2. Mıknatıslanma yöntemi kullanılarak elektrik anahtarı nasıl tasarlanabilir?
3. Mikrodenetleyiciler, diğer çevre birimleri ile (örneğin WiFi modülü, EEPROM vb.) hangi yöntemlerle haberleşir?

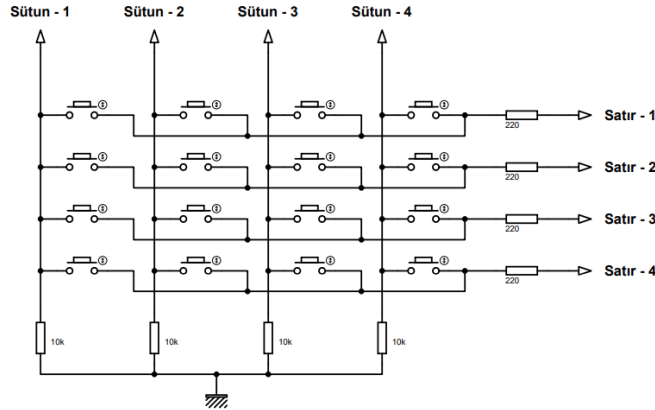
3.1. MİKRODENETLEYİCİ İLE TUŞ TAKIMINDAN VERİ OKUMA

Sayısal sistemlerde kullanıcılar genellikle veri girişini **tuş takımı** diğer ifade ile **keypad (klavye)** adı verilen araçlar ile gerçekleştirir. Tuş takımları butonlar ile oluşturulabileceği gibi hazır tuş takımları da piyasada bulunmaktadır. Görsel 3.1'de piyasada hazır bulunan tuş takımı görseli verilmiştir.



Görsel 3.1: Tuş takımı

Tuş takımları isimlendirilirken 4x4, 4x3 gibi ifadeler kullanılır. Bu gösterimdeki ilk sayı, tuş takımında bulunan satır sayısını, ikinci sayı ise sütun sayısını belirtir. Görsel 3.2'de butonlar ile yapılmış 4x4 bir tuş takımı görülmektedir. Butonların bir uçları bulundukları satırın ortak ucuna, diğer ucu ise bulundukları sütunun ortak uçlarına bağlanır. Bu bağlantı ile tuş takımının sütun ve satırları oluşturmaktadır.



Görsel 3.2: Butonlarla yapılmış 4x4 tuş takımı

Tuş takımında hangi butona basıldığını bulmak için **tarama yöntemi** kullanılır. Tuş takımının sütunları giriş, satırları ise çıkış olarak belirlenir. Tuş takımlarının sütunlarında hep lojik 0 [Şase (GND)] vardır. Hangi butona basıldığını tespit etmek için önce 1. satıra lojik 1 (+ 5V) verilir. Diğer satırlara ise lojik 0 uygulanır. Sonra sütunlar sırası ile okunur. Hangi sütunda lojik 1 varsa o satıra ait sütundaki tuşa basılmıştır. İlk satırda basılan tuş yoksa tüm sütunlarda lojik 0 okunacaktır. Diğer satırlara da sırası ile lojik 1 konumuna alınarak sütunlar sırası ile okunur. Bu işlemler, sütunların herhangi birinden lojik 1 okunana kadar devam eder.

Butona basıldığında ve buton bırakıldığında kısa süreli parazit (ark) oluşur. Tuşa bir kez basılmasına rağmen birden fazla basılmış ya da çekilmiş gibi durum oluşur. Bu duruma **tuş sıçraması (key debounce)** adı verilir. Tuş sıçramasını engellemek için 15-20 ms gecikme oluşturulmalıdır.



UYGULAMA

Adı:	Tuş Takımı Kontrolü	No.: 3.1
Amaç:	Mikrodenetleyici ile tuş takımından veri okumak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda B portuna, butonlardan yapılmış 4x4 tuş takımı bağlanmış mikrodenetleyicinin; D0, D1, D2 ve D3 pinlerine bağlı olan LED’lerde, tuş takımında basılan tuşların binary değer karşılıklarını gösteren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.1’de belirtilmiştir.

Tablo 3.1: 3.1 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Buton		16 adet
Direnç	220 Ω	8 adet
Direnç	10 k Ω	4 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	30 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **TUS_TAKIMI_UYGULAMASI_1** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#define sutun1 PORTB0 //sutun1 ifadesi PORTB0 ifadesine eşitlendi.
#define sutun2 PORTB1 //sutun2 ifadesi PORTB1 ifadesine eşitlendi.
#define sutun3 PORTB2 //sutun3 ifadesi PORTB2 ifadesine eşitlendi.
```



```

#define sutun4 PORTB3          //sutun4 ifadesi PORTB3 ifadesine eşitlendi.
char basilan_tus=0;

/*****Tuş takımı tarama fonksiyonu*****/
char tus_oku()                //tus_oku() fonksiyonu tanımlandı.
{
    PORTB = 0b00010000;       //PORTD=0x08; // 1. satır lojik 1 yapıldı.

    if (PINB&(1<<sutun1))      //1. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =1;
    }
    if (PINB&(1<<sutun2))      //2. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =2;
    }
    if (PINB&(1<<sutun3))      //3. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =3;
    }

    if (PINB&(1<<sutun4))      //4. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =0xA;
    }

    PORTB = 0b00100000;       //PORTD=0x08; // 2. satır lojik 1 yapıldı.

    if (PINB&(1<<sutun1))      //1. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =4;
    }
    if (PINB&(1<<sutun2))      //2. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =5;
    }
    if (PINB&(1<<sutun3))      //3. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =6;
    }
    if (PINB&(1<<sutun4))      //4. sütun okunuyor.
    {
        _delay_us(20);
        basilan_tus =0xB;
    }

    PORTB = 0b01000000;       //PORTD=0x08; // 3. satır lojik 1 yapıldı.

```

```

if (PINB&(1<<sutun1))                //1. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =7;
}
if (PINB&(1<<sutun2))                //2. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =8;
}
if (PINB&(1<<sutun3))                //3. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =9;
}
if (PINB&(1<<sutun4))                //4. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =0xC;
}

PORTB = 0b10000000;                  //PORTD=0x08; // 4. satır lojik 1 yapıldı.

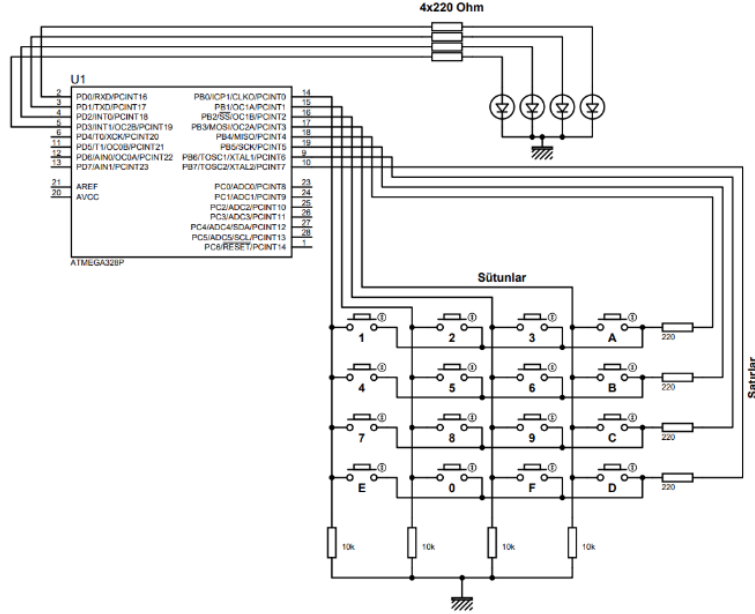
if (PINB&(1<<sutun1))                //1. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =0xE;
}
if (PINB&(1<<sutun2))                //2. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =0;
}
if (PINB&(1<<sutun3))                //3. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =0xF;
}
if (PINB&(1<<sutun4))                //4. sütun okunuyor.
{
    _delay_us(20);
    basilan_tus =0xD;
}
return basilan_tus;                  //Fonksiyon basılan tuş değerini geri döndürdü.
}

int main(void)                        //Ana Fonskiyon
{
    DDRB=0xF0;                        //Sütunlar giriş, satırlar çıkış yapıldı.
    DDRD=0xFF;                        // D portu çıkış olarak ayarlandı.
    while (1)                         //Sonsuz döngü
    {
        PORTD = tus_oku();            //Basılan tuş değeri D portuna aktarıldı.
    }
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

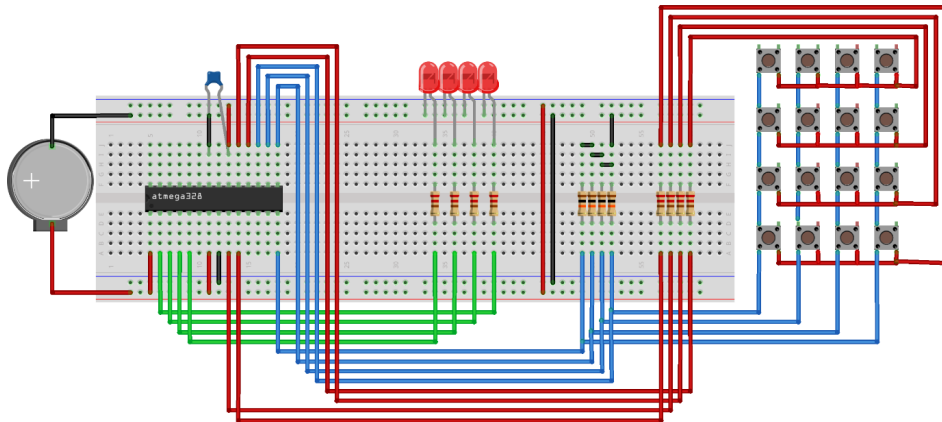
5. Adım: Devre simülasyon programında Görsel 3.3'te verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.3: 3.1 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.4'te verildiği gibi kurunuz.



Görsel 3.4: 3.1 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

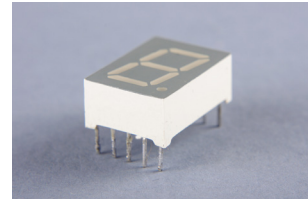
Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		

3.2. MİKRODENETLEYİCİ İLE DISPLAY KONTROLÜ

Günümüzde kullandığımız harf, rakam ve sembolleri göstermek için kullanılan bir gövdeye yerleştirilmiş LED gruplarına display (gösterge) adı verilir.

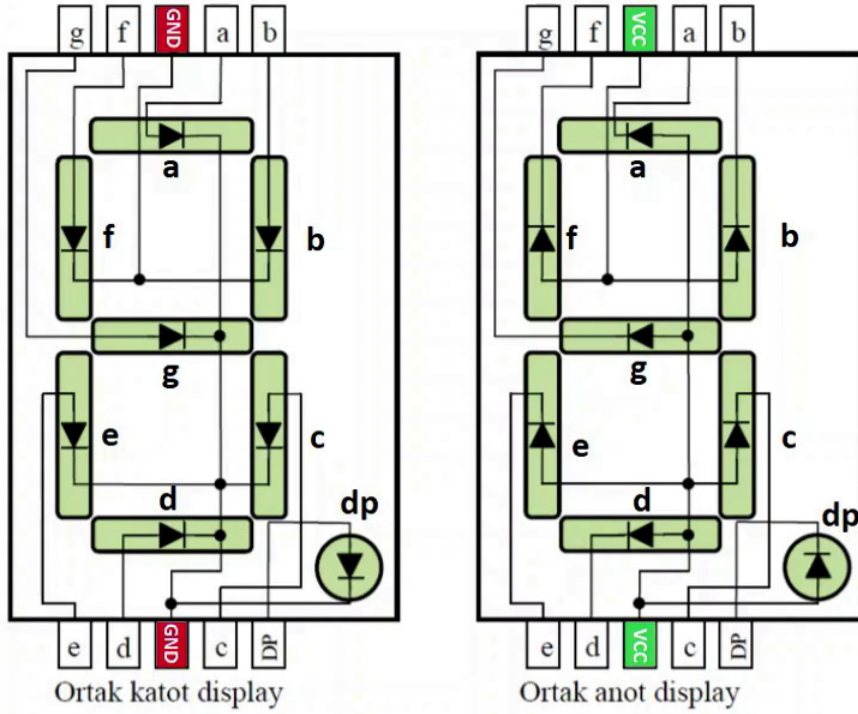
3.2.1. 7 Segment Display (7 Parçalı Gösterge)

7 parçalı gösterge olarak adlandırılan displayler günlük hayatta birçok alanda karşımıza çıkmaktadır. 7 parçalı göstergelerin yapısında yatayda 3 adet dikeyde 4 adet olmak üzere toplamda 7 adet LED bulunmaktadır (Görsel 3.5). Ayrıca 7 segment displaylerde ondalıklı sayıları gösterebilmek için bir adet dp LED'i bulunmaktadır. Bu LED'lerin farklı kombinasyonlarda aktif edilmesi ile gösterge üzerinde harf, rakam ve özel semboller oluşturulabilir. 7 parçalı göstergeler, sensörlerden alınan verilerin mikrodenetleyici tarafından işlenerek anlamlı bir veri olarak harf, rakam ve sembollerle gösterilmesini sağlarlar.



Görsel 3.5: Displayin görünüşü

7 parçalı göstergeler ortak katot ve ortak anot olmak üzere iki türde üretilir. 7 parçalı göstergede bulunan LED'lerin anot bacakları tek noktada birleştirilmiş ise bu göstergelere **ortak anot display** veya göstergede bulunan LED'lerin katot bacakları tek bir noktada birleştirilmiş ise bu göstergelere **ortak katot display** adı verilir.



Görsel 3.6: Ortak katot display (solda), ortak anot display (sağda)

Bu displaylerin bacak bağlantıları her iki gösterge için aynıdır. Sadece ortak uçlar ortak katot displayde GND yani eksi (-) ile beslenmeli, ortak anot displayde ise V_{CC} yani artı (+) ile beslenmelidir. Görsel 3.6'da görüldüğü gibi her parça bir LED yapısına sahiptir. Aktif etmek istediğiniz LED'i doğru yönde bağladığınızda o LED ışık yayacaktır. Aktif etmek istediğiniz LED'in segment uçlarına doğru besleme yaparak istenilen şekil gösterge üzerinde gösterilebilir. Örneğin ortak katot displayde 3 rakamını göstermek için a, b, c, d ve g segmentleri aktif edilmelidir. Bunun için ortak uç olan COM ucuna GND (-) bağlantısı yapılırken a, b, c, d ve g segment uçlarına V_{CC} (+) bağlantısı yapılmalıdır. Ortak anot displayde 7 rakamını göstermek istediğimizde gösterge üzerinde yer alan a, b ve c segmentleri aktif edilmelidir. Bunu için ortak uç olan COM ucuna V_{CC} (+) bağlantısı yapılırken a, b ve c segment uçlarına ise GND (-) bağlantısı yapılmalıdır.

Tablo 3.2'de ortak anot ve ortak katot displayde gösterilmek istenen noktasız karakterler için hangi verinin display bacaklarına gönderilmesi gerektiği bilgisi verilmiştir.

Karakter	Ortak Anot Display									Ortak Katot Display								
	dp	g	f	e	d	c	b	a	HEX	dp	g	f	e	d	c	b	a	HEX
0	0	0	1	1	1	1	1	1	0x3F	1	1	0	0	0	0	0	0	0xC0
1	0	0	0	0	0	1	1	0	0x06	1	1	1	1	1	0	0	1	0xF9
2	0	1	0	1	1	0	1	1	0x5B	1	0	1	0	0	1	0	0	0xA4
3	0	1	0	0	1	1	1	1	0x4F	1	0	1	1	0	0	0	0	0xB0
4	0	1	1	0	0	1	1	0	0x66	1	0	0	1	1	0	0	1	0x99
5	0	1	1	0	1	1	0	1	0x6D	1	0	0	1	0	0	1	0	0x92
6	0	1	1	1	1	1	0	0	0x7C	1	0	0	0	0	0	1	1	0x83
7	0	0	0	0	0	1	1	1	0x07	1	1	1	1	1	0	0	0	0xF8
8	0	1	1	1	1	1	1	1	0x7F	1	0	0	0	0	0	0	0	0x80
9	0	1	1	0	1	1	1	1	0x6F	1	0	0	1	0	0	0	0	0x90
A	0	1	1	1	0	1	1	1	0x77	1	0	0	0	1	0	0	0	0x88
B	0	1	1	1	1	1	0	0	0x7C	1	0	0	0	0	0	1	1	0x83
C	0	0	1	1	1	0	0	1	0x39	1	1	0	0	0	1	1	0	0xC6
D	0	1	0	1	1	1	1	0	0x5E	1	0	1	0	0	0	0	1	0xA1
E	0	1	1	1	1	0	0	1	0x79	1	0	0	0	0	1	1	0	0x86
F	0	1	1	1	0	0	0	1	0x71	1	0	0	0	1	1	1	0	0x8E

Uygulamalarınızda noktasız karakterler kullanılacak ise Tablo 3.2'deki veriler displayin bağlı olduğu porta gönderilmelidir. Noktalı karakter kullanılacak ise dp segmentine gönderilen bit değerleri değiştirilerek kullanılmalıdır.



UYGULAMA

Adı:	0-9 Sayıcı	No.: 3.2
Amaç:	Mikrodenetleyici ile display kontrolünü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda; B portuna bağlı 7 segment göstergede, 0'dan 9'a kadar olan rakamları birer saniye aralıklarla gösteren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.3'te belirtilmiştir.

Tablo 3.3: 3.2 No.lu uygulama için Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Display	7 Segment Display (Cathode)	1 adet
Direnç	220 Ω	7 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	25 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio'da "File" menüsünden "New Project" seçeneğini seçiniz. Görsel 2.62'de görülen "GCC C Executable Project" seçeneğini seçtikten sonra "Name" kısmına **0_9_SAYICI** yazınız.

2. Adım: Görsel 2.63'te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

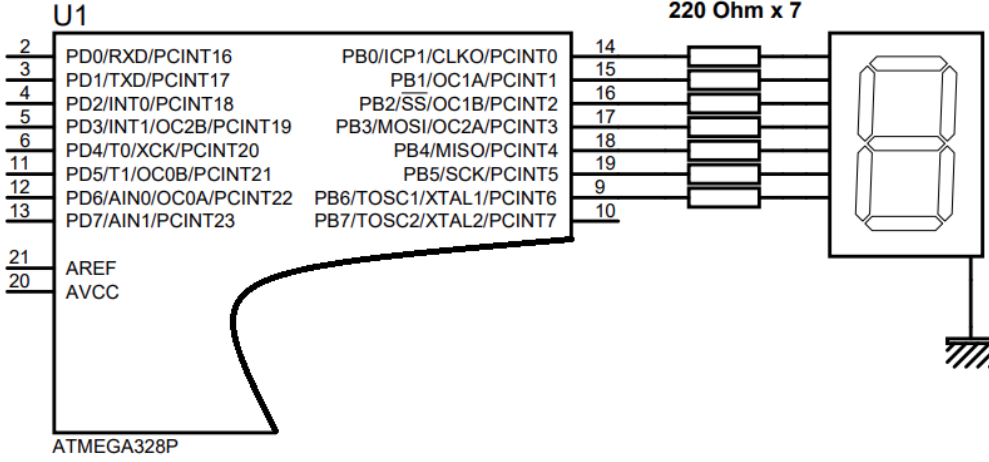
3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <util/delay.h>
#include <avr/io.h>
int main(void)
{
    DDRB=0xFF;           // B portunun tüm pinleri çıkış olarak ayarlandı.
    int sayilar[] = {
        0b00111111,      // 0 Rakamı (0x3F)
        0b00000110,      // 1 Rakamı (0x06)
        0b01011011,      // 2 Rakamı (0x5B)
        0b01001111,      // 3 Rakamı (0x4F)
        0b01100110,      // 4 Rakamı (0x66)
        0b01101101,      // 5 Rakamı (0x6D)
        0b01111100,      // 6 Rakamı (0x7C)
        0b00000111,      // 7 Rakamı (0x07)
        0b01111111,      // 8 Rakamı (0x7F)
        0b01101111};     // 9 Rakamı (0x6F)

    while (1)
    {
        for(int sayac=0;sayac<10;sayac++)    // 0-9 arası sayan döngü
        {
            PORTB=sayilar[sayac];           // PORTB kaydedicisine dizi elemanlarını gönder.
            _delay_ms(1000);                 // 1 saniye gecikme
        }
    }
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

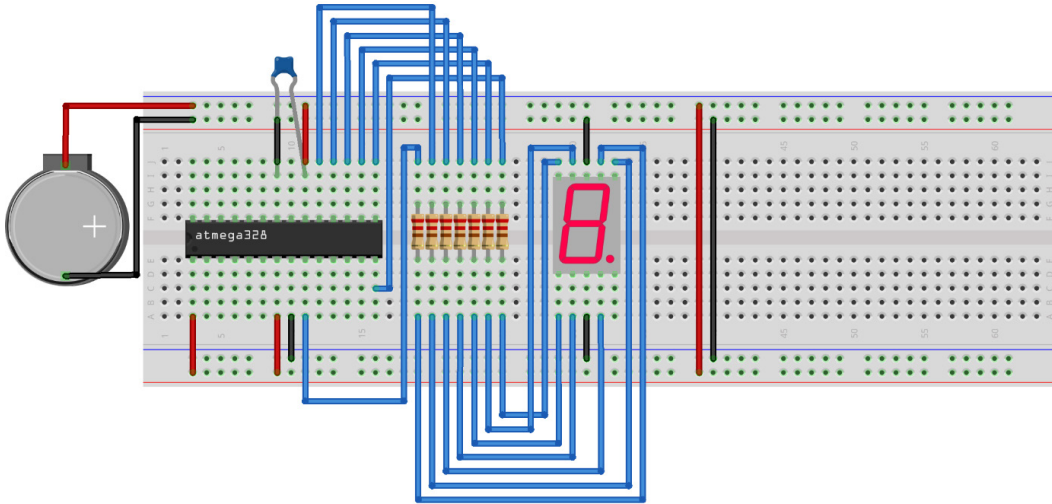
5. Adım: Devre simülasyon programında Görsel 3.7’de verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.7: 3.2 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.8’de verildiği gibi kurunuz.



Görsel 3.8: 3.2 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

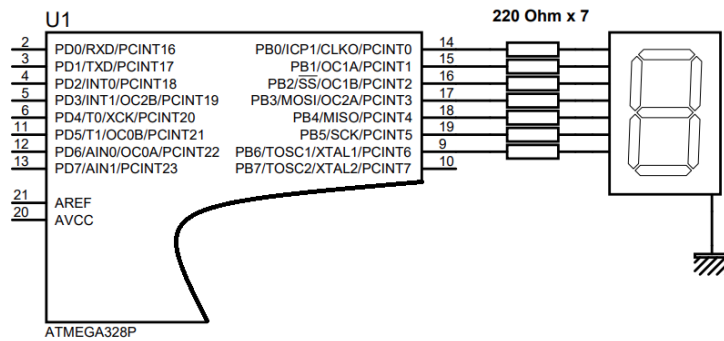
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

B portuna bağlı 7 segment göstergede, 0'dan 9'a kadar olan rakamları birer saniye aralıklar ile önce 0-9 yönünde (ileri) daha sonra 9-0 (geri) yönünde ileri-geri sayan devreyi (Görsel 3.9) iş sağlığı ve güvenliği kurallarını dikkate alarak oluşturunuz ve gömülü yazılımını kodlayınız.



Görsel 3.9: Uygulama devresi



UYGULAMA

Adı:	Buton ile İleri-Geri Sayıcı	No.: 3.3
Amaç:	Mikrodenetleyici ile display kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda; B portuna bağlı 7 segment göstergede, 0'dan 9'a kadar olan rakamları PD2 pinine bağlı butonla ileri (0-9) ve PD3 pinine bağlı butonla geri (0-9) saydıran devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.4'te belirtilmiştir.

Tablo 3.4: 3.3 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Display	7 Segment Display (Cathode)	1 adet
Direnç	220 Ω	9 adet
Direnç	10 k Ω	2 adet
Kondansatör	100 nF	1 adet
Buton		2 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	25 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio'da "File" menüsünden "New Project" seçeneğini seçiniz. Görsel 2.62'de görülen "GCC C Executable Project" seçeneğini seçtikten sonra "Name" kısmına **ILERI_GERI_SAYICI** yazınız.

2. Adım: Görsel 2.63'te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <util/delay.h>
#include <avr/io.h>
int main(void)
{
    DDRD=0x00;           // D portunun tüm pinleri giriş olarak ayarlandı.
    DDRB=0xFF;           // B portunun tüm pinleri çıkış olarak ayarlandı.
    int sayac =0;
```

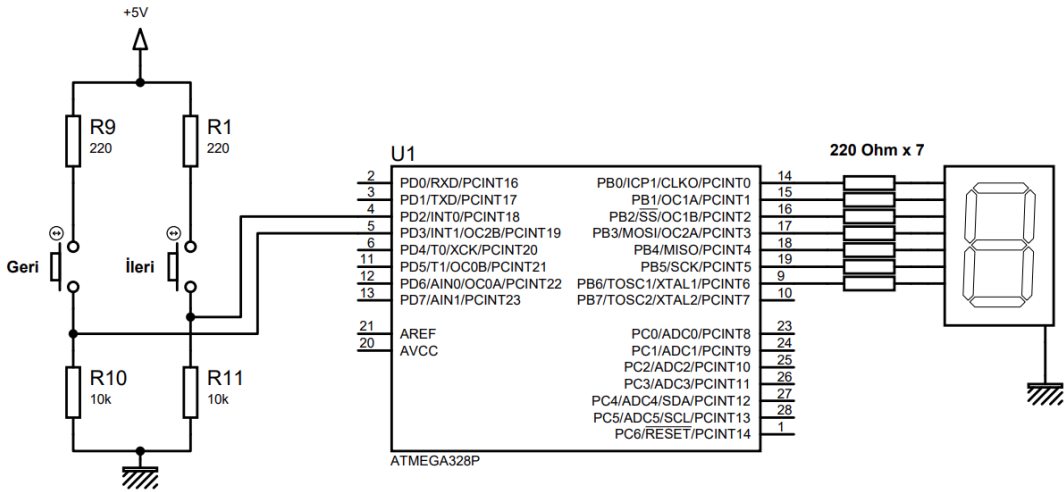
```

int sayilar[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x6F};
PORTB=sayilar[sayac];           // Displaye 0 (sıfır) değeri yazıldı.
while (1)
{
    _delay_ms(200);             // 200 mili saniye gecikme oluştur.
    if(PIND&(1<<PORTD2))        // PD2 pinine bağlı butona (İLERİ) basıldı ise;
    {
        if(sayac<9)             // 9 değerinin üzerinde taşma olmamalı.
        sayac++;
        PORTB=sayilar[sayac];    //Displaye gerekli değeri yaz.
    }
    if(PIND&(1<<PORTD3))        // PD3 pinine bağlı butona (GERİ) basıldı ise
    {
        if(sayac>0)             // 0 değerinin altına taşma olmamalı.
        sayac--;
        PORTB=sayilar[sayac];    //Displaye gerekli değeri yaz.
    }
}
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

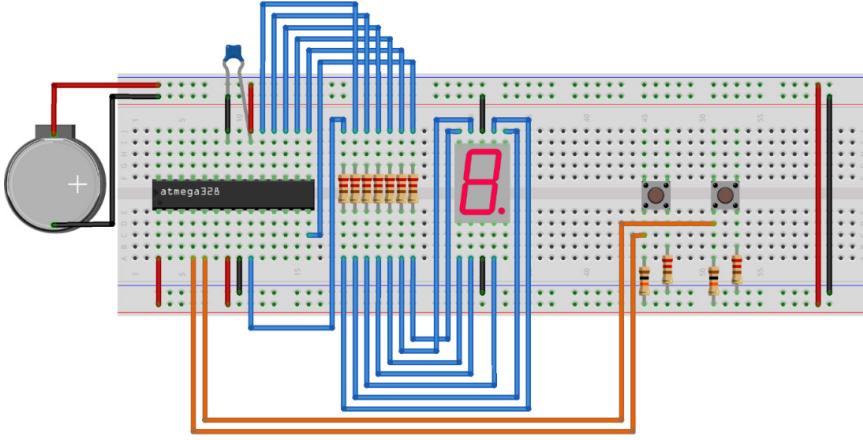
5. Adım: Devre simülasyon programında Görsel 3.10'da verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.10: 3.3 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.11'de verildiği gibi kurunuz.



Görsel 3.11: 3.3 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

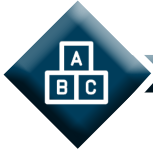
11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



UYGULAMA

Adı:	Yukarı 00-99 Sayıcı	No.: 3.4
Amaç:	Mikrodenetleyici ile display kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda; B ve D portlarına bağlı 7 segment göstergelerde, 0'dan 99'a kadar olan rakamları, birer saniye aralıklarla yukarı saydıran devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.5'te belirtilmiştir.

Tablo 3.5: 3.4 No.lu Uygulama İçin Malzeme Listesi

Adı	Özellği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Display	7 Segment Display (Cathode)	2 adet
Direnç	220 Ω	16 adet
Direnç	10 k Ω	2 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	45 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **ILERI_GERI_SAYICI** yazınız.

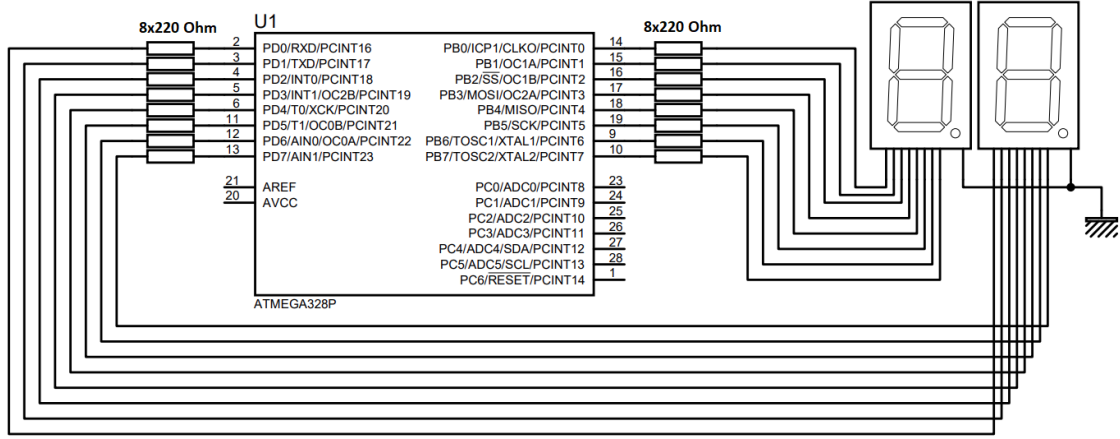
2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <util/delay.h>
#include <avr/io.h>
int main(void)
{
    DDRB=0xFF;           // B portunu çıkış olarak ayarla.
    DDRD=0xFF;           // D portunu çıkış olarak ayarla.
    int sayilar[] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7C,0x07,0x7F,0x6F};
    int sayac=0;          // 0-99 sayacak sayaç değişkeni
    int birler=0;         // Birler basamağını tutacak değişken
    int onlar=0;          // Onlar basamağını tutacak değişken
    PORTB=sayilar[0];     // B portuna 0 (sıfır) değeri yazıldı.
    PORTD=sayilar[0];     // D Portuna 0 (sıfır) değeri yazıldı.
    while (1)
    {
        _delay_ms(1000); // 1 saniye gecikme
        sayac++;         // sayaç bir arttırıldı.
        onlar=sayac/10;   // onlar basamağındaki rakam hesaplandı.
        birler=sayac%10;  // birler basamağındaki rakam hesaplandı.
        if (sayac>99)     // sayac 99 dan büyük ise taşma yaşanmamalı
        {
            sayac=0;      // sayac sıfırlandı.
            birler=0;      // birler sıfırlandı.
            onlar=0;       // onlar sıfırlandı.
        }
        PORTB=sayilar[onlar]; // B portuna onlar basamağındaki rakam yazıldı.
        PORTD=sayilar[birler]; // D portuna birler basamağındaki rakam yazıldı.
    }
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

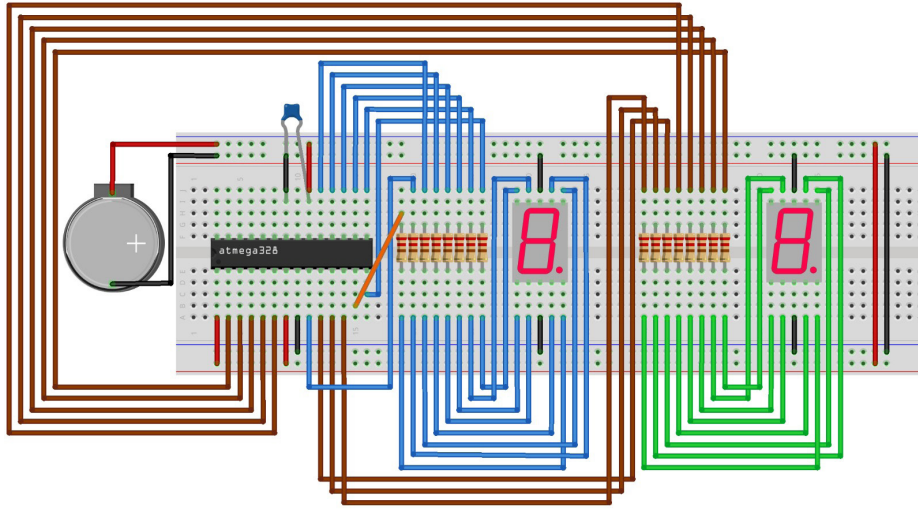
5. Adım: Devre simülasyon programında Görsel 3.12’de verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.12: 3.4 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.13’te verildiği gibi kurunuz.



Görsel 3.13: 3.4 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

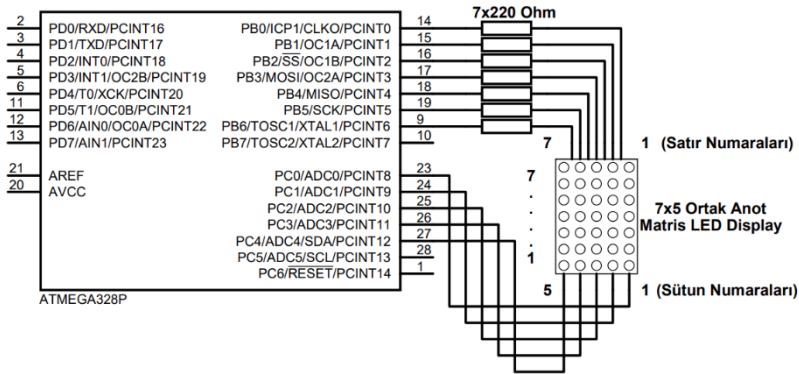
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

B ve D portlarına bağlı 7 parçalı göstergelerde, 0'dan 99'a kadar olan rakamları birer saniye aralıklarla yukarı-aşağı saydıran devreyi (Görsel 3.14) iş sağlığı ve güvenliği kurallarını dikkate alarak oluşturunuz ve gömülü yazılımını kodlayınız.

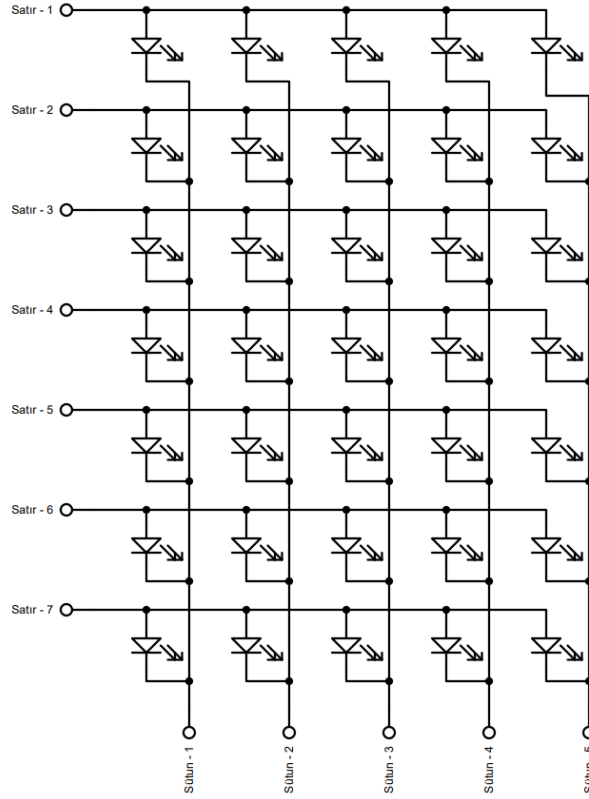


Görsel 3.19: Uygulama devresi

3.2.2. Matris LED Display

Görüntüleme amacı ile kullanılan LED veya 7 segment display, her uygulama için yeterli olmayabilir. Günümüzde görüntüleme işlemlerinde daha çok LED'lerin matris yapısında bağlanarak elde edilen matris LED displayler kullanılmaktadır. Bu displaylerin satır ve sütunlarını LED'ler oluşturmaktadır. Harf, rakam ya da sembolün görüntüsünü matris LED display üzerinde

oluşturmak istediğimizde uygun satır ve sütunlardaki LED'ler aktif edilmelidir. Piyasada 5x7 (5 sütun, 7 satır), 8x8 (8 sütun, 8 satır) boyutlarında matris LED displayler bulunmaktadır. Görsel 3.15'te 5x7 matris LED displayin iç bağlantısı ve dış görüntüsü verilmiştir.

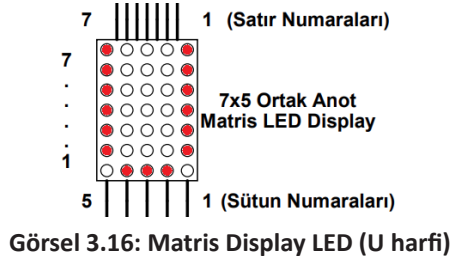


Görsel 3.15: 5x7 Matris LED Display iç yapısı

Matris LED displaylerde satır ve sütunları temsil eden bilgi uçları bulunmaktadır. Matris LED'lerde istenilen karakteri göstermek için tarama metodu kullanılır. Yani ilk olarak birinci sütun aktif edilir daha sonra ilk sütunda yer alan LED'lerden istenileni yakmak için istenilen satır veya satırlar aktif edilir. Daha sonra ikinci sütun aktif edilir ve ikinci sütunda aktif edilmek istenen LED'lerin bulunduğu satır veya satırlar aktif edilir. Bu işlem tüm sütunlar taranana kadar devam eder. Böylelikle matris LED display üzerinde istenilen karakter oluşturulmuş olur. Tarama metodu ile oluşturulan karakterin insan gözü tarafından algılanabilmesi için tarama işleminin yeterli süre tekrar etmesi gerekmektedir. Piyasada tarama işlemlerini gerçekleştiren entegrelerde mevcuttur.

Piyasada katot sütun ve anot sütun olmak üzere iki çeşit matris LED display bulunmaktadır. Katot sütun matris LED displaylerin sütunlarını aktif etmek için ilgili sütuna lojik 0, satırları aktif etmek için ilgili satıra lojik 1 değeri uygulanır. Anot sütun matris LED displaylerde ise sütunları aktif etmek için ilgili sütuna lojik 1, satırları aktif etmek için ise ilgili satıra lojik 0 değeri uygulanmalıdır.

Örneğin ortak anot matris LED display üzerinde U harfini Görsel 3.16’da verildiği gibi göstermek için oluşturulması gereken kodun algoritması aşağıda anlatılmıştır.



Ortak anot matris LED displaylerde istenilen LED’i yakmak için ilgili sütuna lojik 1 ilgili satıra ise lojik 0 değeri gönderilmelidir.

ALGORİTMA

Görev: Ortak anot matris LED display üzerinde U harfini gösterme

1. Adım: 1. Sütun aktif (C portunun 0 No.lu pinine lojik 1 değeri gönder) (PORTC = 0x01;)

2. Adım: 1. Sütunda aktif edilecek LED’lere karşılık gelen satırlara lojik 0 değeri gönder (PORTB = 0x01;).

3. Adım: Gecikme

4. Adım: 2. Sütun aktif (C portunun 1 No.lu pinine lojik 1 değeri gönder) (PORTC = 0x02;)

5. Adım: 2. Sütunda aktif edilecek LED’lere karşılık gelen satırlara lojik 0 değeri gönder (PORTB = 0xFE;).

6. Adım: Gecikme

7. Adım: 3. Sütun aktif (C portunun 2 No.lu pinine lojik 1 değeri gönder) (PORTC = 0x04;)

8. Adım: 3. Sütunda aktif edilecek LED’lere karşılık gelen satırlara lojik 0 değeri gönder (PORTB = 0xFE;).

9. Adım: Gecikme

10. Adım: 4. Sütun aktif (C portunun 3 No.lu pinine lojik 1 değeri gönder) (PORTC = 0x08;)

11. Adım: 4. Sütunda aktif edilecek LED’lere karşılık gelen satırlara lojik 0 değeri gönder (PORTB = 0xFE;).

12. Adım: Gecikme

13. Adım: 5. Sütun aktif (C portunun 4 No.lu pinine lojik 1 değeri gönder) (PORTC = 0x10;)

14. Adım: 5. Sütunda aktif edilecek LED’lere karşılık gelen satırlara lojik 0 değeri gönder (PORTB = 0x01;).

15. Adım: Gecikme



UYGULAMA

Adı:	Matris LED Display	No.: 3.5
Amaç:	Mikrodenetleyici ile matris LED display kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, mikrodenetleyicinin B ve C portlarına bağlı olan matris LED displayde, U harfini gösteren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.6’da belirtilmiştir.

Tablo 3.6: 3.5 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Display	7x5 Ortak Anot Matris LED Display	1 adet
Direnç	220 Ω	7 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	45 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **MATRIS_LED_SURME** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRB=0xFF;           // B portunu çıkış olarak ayarla.
    DDRC=0xFF;           // C portunu çıkış olarak ayarla.
```

```

while (1)
{
    PORTC=0x01;           // 1. sütunu aktif et.
    PORTB=0x01;           // ilgili satırları aktif et.
    _delay_ms(5);         // 5 ms gecikme

    PORTC=0x02;           // 2. sütunu aktif et.
    PORTB=0xFE;           // ilgili satırları aktif et.
    _delay_ms(5);         // 5 ms gecikme

    PORTC=0x04;           // 3. sütunu aktif et.
    PORTB=0xFE;           // ilgili satırları aktif et.
    _delay_ms(5);         // 5 ms gecikme

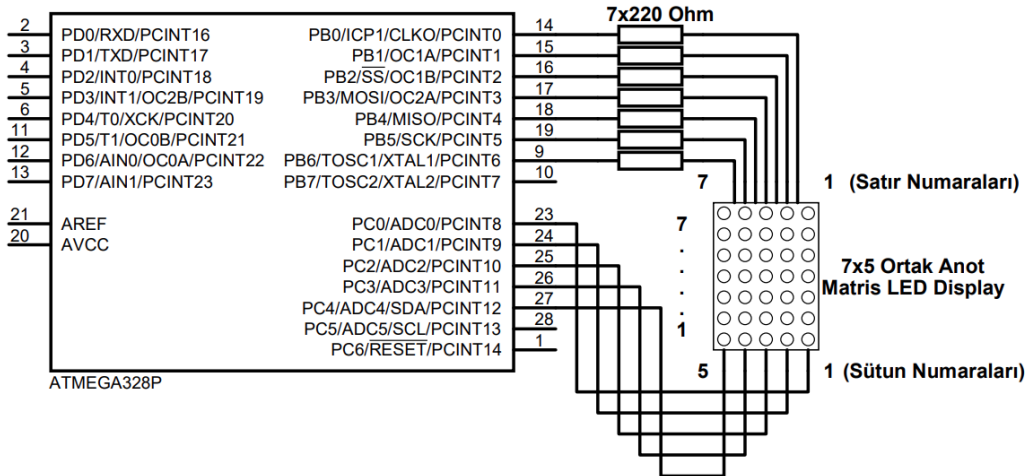
    PORTC=0x08;           // 4. sütunu aktif et.
    PORTB=0xFE;           // ilgili satırları aktif et.
    _delay_ms(5);         // 5 ms gecikme

    PORTC=0x10;           // 5. sütunu aktif et.
    PORTB=0x01;           // ilgili satırları aktif et.
    _delay_ms(5);         // 5 ms gecikme
}
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

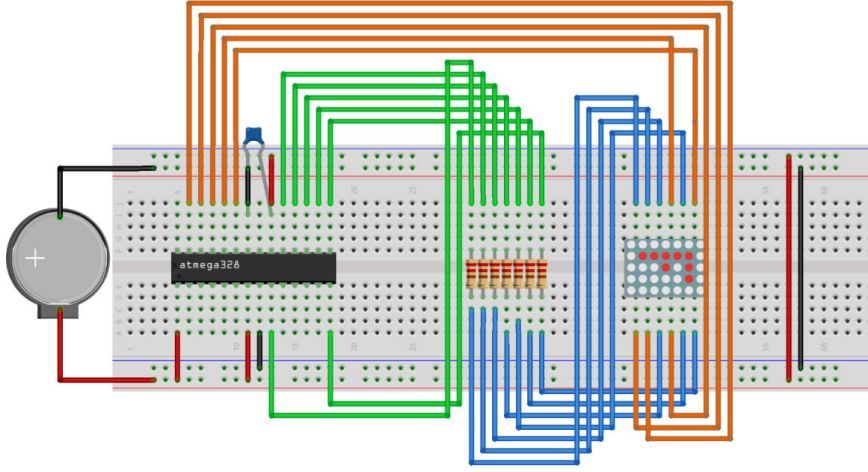
5. Adım: Devre simülasyon programında Görsel 3.17’de verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.17: 3.5 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.18’de verildiği gibi kurunuz.



Görsel 3.18: 3.5 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

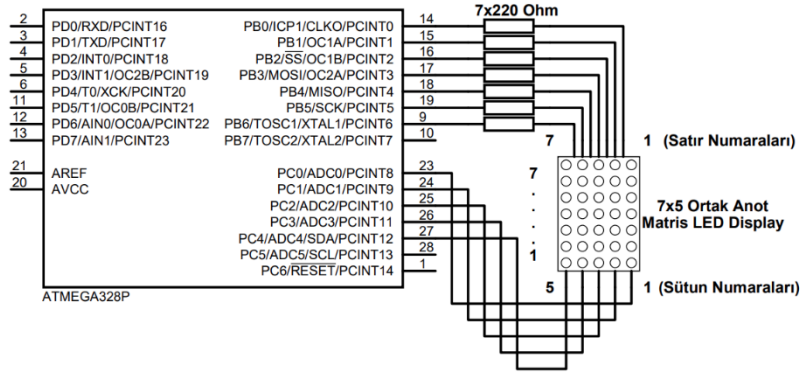
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

Mikrodenetleyicinin B ve C portlarına bağlı olan matris LED displayde K harfini gösteren devreyi (Görsel 3.19) iş sağlığı ve güvenliği kurallarını dikkate alarak oluşturunuz ve gömülü yazılımını kodlayınız.



Görsel 3.19: Uygulama devresi



UYGULAMA

Adı:	Matris LED Display ile Alfabe Uygulaması	No.: 3.6
Amaç:	Mikrodenetleyici ile matris LED display kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek adıgıer sayfada verilen adımlar doğrultusunda, mikrodenetleyicinin B ve C portlarına bağlı olan matris LED displayde, alfabede bulunan harfleri birer saniye aralıkla gösteren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.7’de belirtilmiştir.

Tablo 3.7: 3.6 No.lu Uygulama İçin Malzeme Listesi

Adı	Özellği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Display	7x5 Ortak Anot Matris LED Display	1 adet
Direnç	220 Ω	7 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	45 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen” GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **ALFABE** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
    DDRB=0xFF;    //B portu çıkış olarak ayarlandı.
    DDRC=0xFF;    //C portu çıkış olarak ayarlandı.
    // Alfabe matrisinin oluşturulması:
    int alfabe[28][5] = {{0xC0,0x37,0x37,0x37,0xC0},           //A
                        {0x49,0x36,0x36,0x36,0x00},           //B
                        {0x3E,0x3E,0x3E,0x3E,0x41},           //C
                        {0x41,0x3E,0x3E,0x3E,0x00},           //D
                        {0x3E,0x36,0x36,0x36,0x00},           //E
                        {0x3F,0x37,0x37,0x37,0x00},           //F
                        {0x31,0x36,0x36,0x3E,0x41},           //G
                        {0x00,0x77,0x77,0x77,0x00},           //H
                        {0xFF,0xFF,0x00,0xFF,0xFF},           //I
                        {0x01,0xFE,0xFE,0xFE,0xFD},           //J
                        {0x3E,0x5D,0x6B,0x77,0x00},           //K
                        {0xFE,0xFE,0xFE,0xFE,0x00},           //L
                        {0x00,0x5F,0x6F,0x5F,0x00},           //M
                        {0x00,0xF7,0x6F,0x5F,0x00},           //N
                        {0x41,0x3E,0x3E,0x3E,0x41},           //O
                        {0x4F,0x37,0x37,0x37,0x00},           //P
                        {0x4E,0x35,0x33,0x37,0x00},           //R
                        {0x30,0x36,0x36,0x36,0x06},           //S
                        {0x3F,0x3F,0x00,0x3F,0x3F},           //T
                        {0x01,0xFE,0xFE,0xFE,0x01},           //U
                        {0x03,0xFD,0xFE,0xFD,0x03},           //V
                        {0x3F,0x5F,0x60,0x5F,0x3F},           //Y
                        {0x1E,0x2E,0x36,0x3A,0x3C},           //Z
                        };

    while (1)
    {
        for(int 25ayaç=0; 25ayaç<23; 25ayaç++) //Alfabe dizisinde dolaşmak için döngü
        {
            for(int i=0;i<16;i++) //Bir karakteri, 16 kez dislaye bas (Gözün
                //algılayabilmesi için).
            {
```

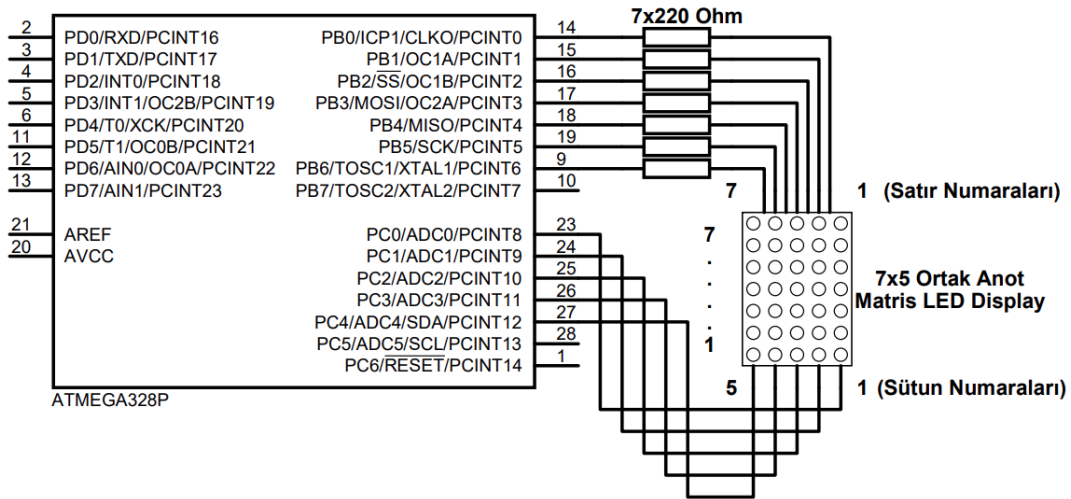
```

PORTC =0x01;           //1. Sütunu aktif et.
PORTB=alfabe[ 25ayaç][0]; // Sıradaki harfin 1. Sütununu bas.
_delay_ms(5);           // Gecikme
PORTC =0x02;           //2. Sütunu aktif et.
PORTB=alfabe[ 25ayaç][1]; // Sıradaki harfin 2. Sütununu bas.
_delay_ms(5);           // Gecikme
PORTC =0x04;           //3. Sütunu aktif et.
PORTB=alfabe[ 25ayaç][2]; // Sıradaki harfin 3. Sütununu bas.
_delay_ms(5);           // Gecikme
PORTC =0x08;           //4. Sütunu aktif et.
PORTB=alfabe[ 25ayaç][3]; // Sıradaki harfin 4. Sütununu bas.
_delay_ms(5);           // Gecikme
PORTC =0x10;           //5. Sütunu aktif et.
PORTB=alfabe[ 25ayaç][4]; // Sıradaki harfin 5. Sütununu bas.
_delay_ms(5);           // Gecikme
    }
    PORTB=0xFF;           //Matris LED Displayi Temizle.
    _delay_ms(1000);      //1 saniye gecikme
}
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

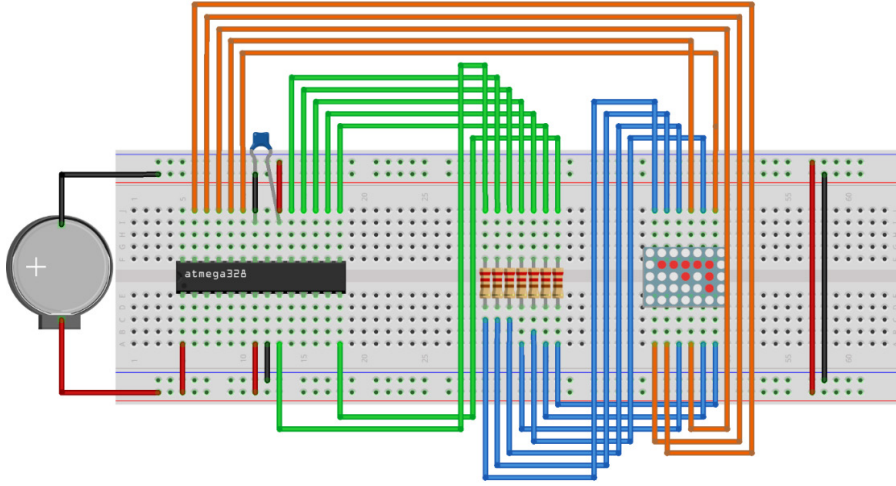
5. Adım: Devre simülasyon programında Görsel 3.20’de verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.20: 3.6 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.21’de verildiği gibi kurunuz.



Görsel 3.21: Uygulama 3.6 için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

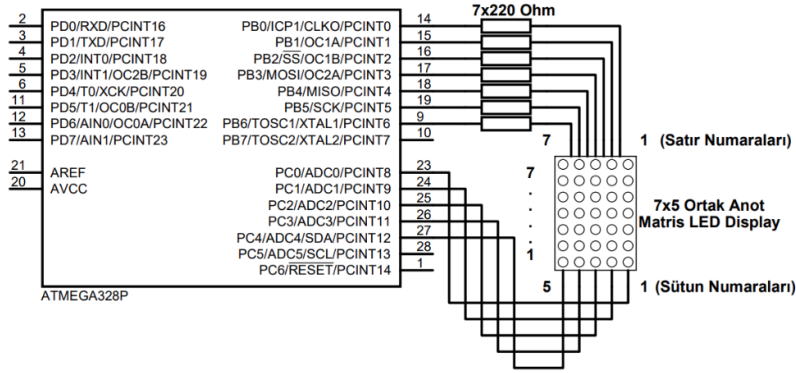
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

Mikrodenetleyicinin B ve C portlarına bağlı olan matris LED displayde rakamları birer saniye aralıkla gösteren devreyi (Görsel 3.22) iş sağlığı ve güvenliği kurallarını dikkate alarak oluşturunuz ve gömülü yazılımını kodlayınız.

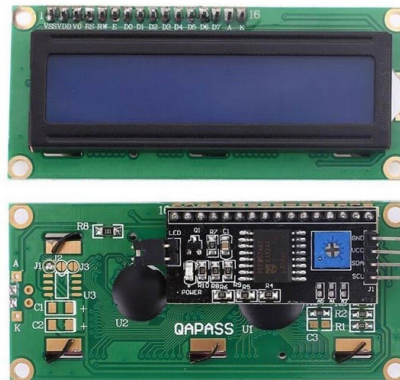


Görsel 3.22: Uygulama devresi

3.2.3. Karakter LCD

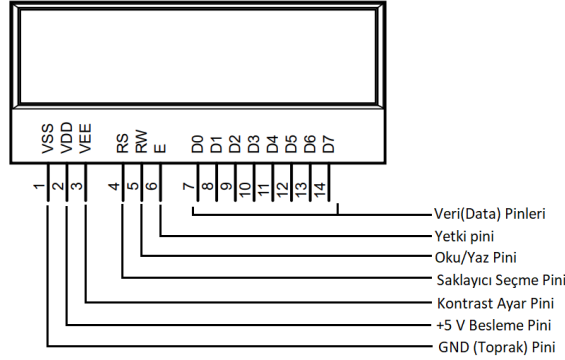
Gömülü sistemlerde elde edilen durum veya parametreleri görüntülemek için LCD'ler kullanılır. LCD, Liquid Crystal Display (Sıvı Kristal Görüntüleme Birimi) açılımının kısaltmasıdır. LCD'ler bize sembol, harf ve karakterleri görüntüleme imkânı sağlar. Piyasada karakter tabanlı ve grafik tabanlı LCD'ler bulunmaktadır.

Karakter tabanlı LCD'ler 5 x 8'lik veya 5 x 10'luk nokta matris hücreden oluşurlar. LCD'lerde karakterlerin gösterilmesi matris LED displayde olduğu gibi tarama mantığı ile sağlanır. Tarama işlemi denetleyici ile yapılmaya çalışılırsa bu işlem denetleyiciyi tamamen meşgul edeceğinden darboğaz oluşur. LCD üzerinde tarama işlemini gerçekleştiren entegreler mevcuttur. LCD'ler satır ve sütun numaralarına göre isimlendirilir. Örneğin 2 x 16 LCD, 2 satır 16 sütundan oluşur. Görsel 3.23'te 2 x 16 LCD'nin ön ve arka panel görüntüsü verilmiştir.



Görsel 3.23: 2x16 LCD ön ve arka panel

Genelde bir karakter LCD'nin 14 pini bulunmaktadır. Bunun yanında bazı LCD'lerde 2 adet arka ışık besleme pini de bulunmaktadır. Yani 16 pinli LCD'lerde iki uç arka ışık beslemesi için + ve – ucu sağlamaktadır. Görsel 3.24'te 2 x 16 LCD'nin pin isimleri verilmiştir.



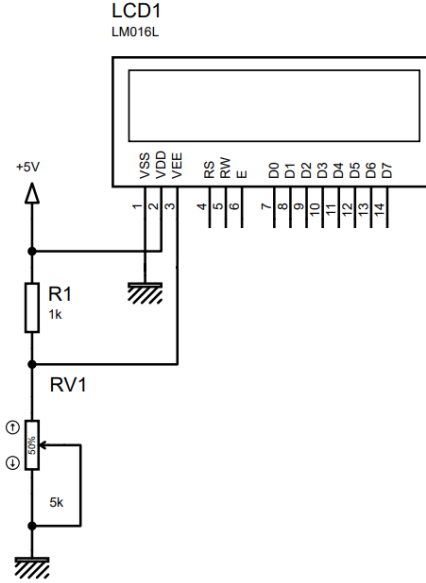
Görsel 3.24: Karakter LCD pin isimleri

Karakter LCD'leri kullanabilmek için pinlerinin görevleri bilinmelidir. LCD pinleri Tablo 3.8'de sırası ile tanıtılmıştır.

Tablo 3.8: LCD Pinleri ve Görevleri

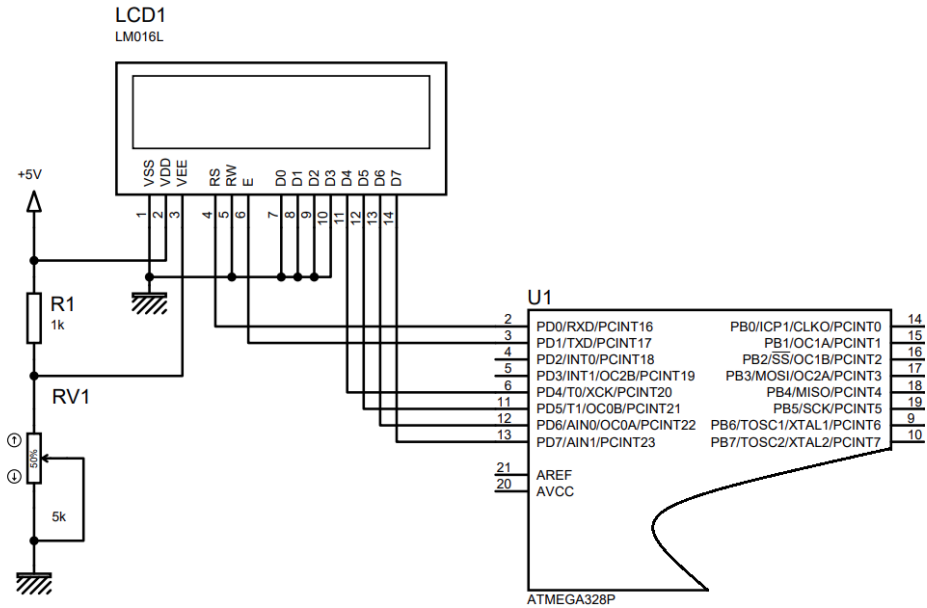
Pin No.	Pin İsmi	Görevi
1	VSS	Şase ucudur. LCD için toprak bağlantısının yapıldığı pindir.
2	VDD	Pozitif besleme ucudur. Bu pine +5 V uygulanmalıdır.
3	VEE	Kontrast ayar ucudur. Görünen karakterlerin parlaklık ayar pinidir.
4	RS	Veri yolundan gelen verinin komut ya da veri bilgisi olup olmadığının belirlendiği pindir. RS = 0 ise D0, D7 arası pinlerden gelen bilgi, komut olarak algılanır. RS = 1 ise D0, D7 arası pinlerden gelen bilgi karakter olarak algılanır.
5	RW	LCD'ye bilgi göndermek ve LCD'den bilgi almak için kullanılan pindir. RW = 0 ise LCD'ye bilgi gönderme yapılır. RW = 1 ise LCD'den bilgi okunur.
6	E	Yetki ucudur. Okuma ya da yazma işleminin gerçekleşebilmesi için yetki ucu önce lojik 1 daha sonra lojik 0 yapılması gerekmektedir. Yani okuma ya da yazma işlemi düşen kenarda gerçekleşir.
7 ila 14	D0 ila D7	Veri uçlarıdır. Bu uçlar ile LCD'ye veri ve komut gönderme işlemi yapılabilir. Bu uçlar ile LCD'den veri alma işlemi de gerçekleştirilir.
15-16	A-K	LCD arka ışıklandırma besleme uçlarıdır. A pini +5 V ile beslenmelidir. K pini ise GND (şase) ile beslenmelidir.

LCD kontrast ayarını isteğe bağlı olarak ayarlamak için VEE pini ile VSS pini arasına Görsel 3.25'te gösterildiği gibi bir potansiyometre bağlanır.



Görsel 3.25: LCD besleme ve kontrast bağlantısı

2x16 LCD'yi kullanabilmek için Görsel 3.26'da verilen bağlantı şemasının uygulanması gerekmektedir.



Görsel 3.26: LCD'nin Denetleyiciye bağlanması

LCD üzerinde denetleyiciden gelen komutları işleyebilmesi için bir adet kontrolcü bulunur. Bu kontrolcü belli komutlara göre çalışır. Tablo 3.9'da kontrolcü komutları verilmiştir.

Tablo 3.9: LCD Kontrolcü Komutları

Display Komutları	
Komut	Açıklama
0 x 01	Bu komut ile display silinir. İmleç 1. Satır 1. Sütuna konumlanır.
0 x 02 – 0 x 03	Bu komut verildiğinde imleç 1. Satır 1. Sütuna konumlanır.
0 x 08	Display kapatılır. İmleç gösterilmez. İmleç yanıp sönmez.
0 x 09	Display kapatılır. İmleç gösterilmez. İmleç yanıp söner.
0 x 0A	Display kapatılır. İmleç gösterilir. İmleç yanıp sönmez.
0 x 0B	Display kapatılır. İmleç gösterilir. İmleç yanıp söner.
0 x 0C	Display açılır. İmleç gösterilmez. İmleç yanıp sönmez.
0 x 0D	Display açılır. İmleç gösterilmez. İmleç yanıp söner.
0 x 0E	Display açılır. İmleç gösterilir. İmleç yanıp sönmez.
0 x 0F	Display açılır. İmleç gösterilir. İmleç yanıp söner.
0 x 10	İmleç bir karakter sola kaydırılır.
0 x 14	İmleç bir karakter sağa kaydırılır.
0 x 18	Display sola kaydırılır.
0 x 1C	Display sağa kaydırılır.
Fonksiyon Ayar Komutları	
Komut	Açıklama
0 x 3C	LCD 1 satır, 5x8 nokta matris, 4 bit iletişim formatında ayarlanır.
0 x 24	LCD 1 satır, 5x10 nokta matris, 4 bit iletişim formatında ayarlanır.
0 x 28	LCD 2 satır, 5x8 nokta matris, 4 bit iletişim formatında ayarlanır.
0 x 2C	LCD 2 satır, 5x10 nokta matris, 4 bit iletişim formatında ayarlanır.
0 x 30	LCD 1 satır, 5x8 nokta matris, 8 bit iletişim formatında ayarlanır.
0 x 34	LCD 1 satır, 5x10 nokta matris, 8 bit iletişim formatında ayarlanır.
0 x 38	LCD 2 satır, 5x8 nokta matris, 8 bit iletişim formatında ayarlanır.
0 x 20	LCD 2 satır, 5x10 nokta matris, 8 bit iletişim formatında ayarlanır.
Giriş Modu Komutları	
Komut	Açıklama
0 x 04	LCD'de gösterilen her karakterden sonra imleç otomatik sola kayar.
0 x 05	LCD'de gösterilen her karakterden sonra imleç ve display otomatik sola kayar.
0 x 06	LCD'de gösterilen her karakterden sonra imleç otomatik sağa kayar.
0 x 07	LCD'de gösterilen her karakterden sonra imleç ve display otomatik sağa kayar.

3.2.3.1. Kişiyi Özel LCD Dosyası Oluşturma

Karakter LCD kullanabilmek için hazır kütüphaneler bulunmaktadır. Hazır kütüphaneler kullanıldığında çok kolay bir şekilde karakter LCD'ler kullanılabilir. Bu konu başlığı altında hazır kütüphaneleri kullanmak yerine yeni fonksiyonlar oluşturularak karakter LCD çalışma mantığının daha iyi kavranması amaçlanmıştır.

Karakter LCD kullanabilmek için gerekli komut ve fonksiyonları oluşturma işlemi aşağıdaki gibidir.

- Karakter LCD için gerekli olan sabitlerin tanımlaması yapılır.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>

#define LCD_Port PORTD // LCD portunu tanımla (PORTD).
#define LCD_DataPin DDRD // 4-bit pinleri tanımla (PORTD4-PORTD7).
#define RS PD0 // RS pinini tanımla.
#define EN PD1 // E pinini tanımla.
```

Yukarıdaki komutlar ile karakter LCD'nin bağlı olduğu pinlerin görev tanımları yapılmış olur. Bu sabitleri, kodlar içinde kullanarak kodun okunurluğu artar.

- Karakter LCD'ye gelen komutları işlemesi için gerekli olan fonksiyon oluşturulur.

```
Void LCD_Komut( unsigned char komut ) // LCD çalışma kodları
{
    LCD_Port = (LCD_Port & 0x0F) | (komut & 0xF0); //Veri portuna komut gönderildi.
    LCD_Port &= ~(1 << RS); //RS=0, gönderilecek bilginin komut olduğu bildirildi.
    LCD_Port |= (1 << EN); //E=1
    _delay_us(1);
    LCD_Port &= ~(1 << EN); //E=0
    _delay_us(200);
    LCD_Port = (LCD_Port & 0x0F) | (komut << 4);
    LCD_Port |= (1 << EN);
    _delay_us(1);
    LCD_Port &= ~(1 << EN);
    _delay_ms(2);
}
```

Okuma ya da yazma işlemi gerçekleştirilmesi için yetki ucunun E önce lojik 1 (`LCD_Port |= (1 << EN);`), sonrada lojik 0 (`LCD_Port &= ~(1 << EN);`) yapılması gerekir. Okuma veya yazma işlemi düşen kenarda gerçekleşir.

- Karakter LCD'yi başlatacak fonksiyon yazılır.

```
void LCD_Baslat (void)
{
    LCD_DataPin = 0xFF;      // LCD pinlerini kontrol et (PORTD4-PORTD7)
    _delay_ms(15);           // LCD etkinleştirilinceye kadar bekle
    LCD_Komut(0x02);         // 4-bit kontrol
    LCD_Komut(0x28);         // Kontrol matris ayarı
    LCD_Komut(0x0c);         // İmleç devre dışı
    LCD_Komut(0x06);         // İmleci taşı
    LCD_Komut(0x01);         // LCD'yi temizle
    _delay_ms(2);            // 2 ms bekle
}
```

LCD, ilk açılışta 15 ms'lik bir kendi kendini başlatma süresine ihtiyaç duyar. Bu süreye **LCD açılış gecikmesi** adı verilir. LCD'ye açılış gecikme süresi içerisinde komut gönderilemez. Komut gönderilse bile LCD bu komutları işleyemez. Bu sebeple LCD programlanırken açılış gecikme süresi hesaba katılarak açılışta en az 15 ms'lik bir gecikme oluşturulmalıdır. Yukarıda verilen kod blokunda `_delay_ms(15);` komutu ile bu gecikme oluşturulmuştur.

LCD'ler data portuna gelen bilgileri karakter karakter ekranda gösterir. Bir karakter katarının tamamı tek seferde data portuna yazılamaz. Bu sebeple karakter katarındaki her bir karakter, sırası ile data portuna gönderilmelidir. Bunu sağlamak için aşağıdaki fonksiyon kullanılabilir.

```
void LCD_Yaz (char *str)      // Karakter dizisini LCD'ye yaz
{
    int i;
    for(i=0; str[i]!=0; i++)
    {
        LCD_Port = (LCD_Port & 0x0F) | (str[i] & 0xF0);
        LCD_Port |= (1 << RS);
        LCD_Port |= (1 << EN);
        _delay_us(1);
        LCD_Port &= ~(1 << EN);
        _delay_us(200);
        LCD_Port = (LCD_Port & 0x0F) | (str[i] << 4);
        LCD_Port |= (1 << EN);
        _delay_us(1);
        LCD_Port &= ~(1 << EN);
        _delay_ms(2);
    }
}
```

Yukarıdaki fonksiyon, aldığı karakter katarının her karakterini tek tek LCD'ye gönderir. Karakter katarının son karakterine kadar dönen bir for döngüsü oluşturularak döngü içerisinde karakter katarının her elemanı tek tek LCD'ye gönderilir.

- İstenildiğinde LCD'nin temizlenmesi için gerekli temizleme fonksiyonunu oluşturulur.

```
void LCD_Temizle()           // LCD temizleme.
{
    LCD_Komut (0x01);        // LCD'yi temizle.
    _delay_ms(2);            // LCD temizlenene kadar bekle.
    LCD_Komut (0x80);        // İmleci 1. satır, 1. sütun pozisyonuna getir.
}
```

Yukarıda verilen fonksiyonlar, projede **main.c** dosyasının bulunduğu dizinde **Include** adında bir dizin oluşturulup içerisine **lcd.h** adında bir dosya açarak içerisine kopyalanmalıdır. Yazılan lcd.h dosyasını projeye dâhil etmek için aşağıdaki komut kullanılır.

```
#include ".\include\lcd.h"
```



UYGULAMA

Adı:	Karakter LCD Uygulaması	No.: 3.7
Amaç:	Mikrodenetleyici ile karakter LCD kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, mikrodenetleyiciye bağlı olan karakter LCD ekranına, "LCD EKRAN UYGULAMASI" yazdıran devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.10'da belirtilmiştir.

Tablo 3.10: 3.7 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LCD	Karakter LCD (16x2)	1 adet
Direnç	10 kΩ	1 adet
Potansiyometre	5 kΩ	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	20 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio'da "File" menüsünden "New Project" seçeneğini seçiniz. Görsel 2.62'de görülen "GCC C Executable Project" seçeneğini seçtikten sonra "Name" kısmına **LCD_UYGULAMASI** yazınız.

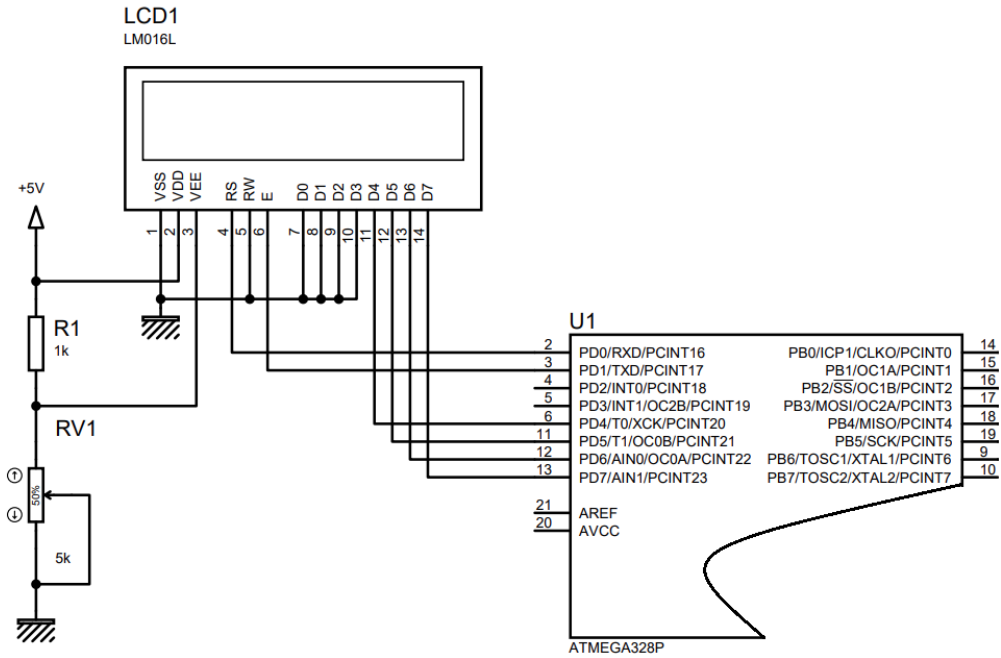
2. Adım: Görsel 2.63'te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include ".\include\lcd.h"           // Projenize lcd.h dosyası dâhil edildi.
int main()
{
    LCD_Baslat();                   // LCD'yi başlat.
    LCD_Yaz("LCD EKRAN");           // LCD'ye "LCD EKRAN" yaz.
    LCD_Komut(0xC0);                // 2. satır, 1. sütuna imleci konumlandır.
    LCD_Yaz("UYGULAMASI");          // LCD'ye "UYGULAMASI" yaz.
    return 0;
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

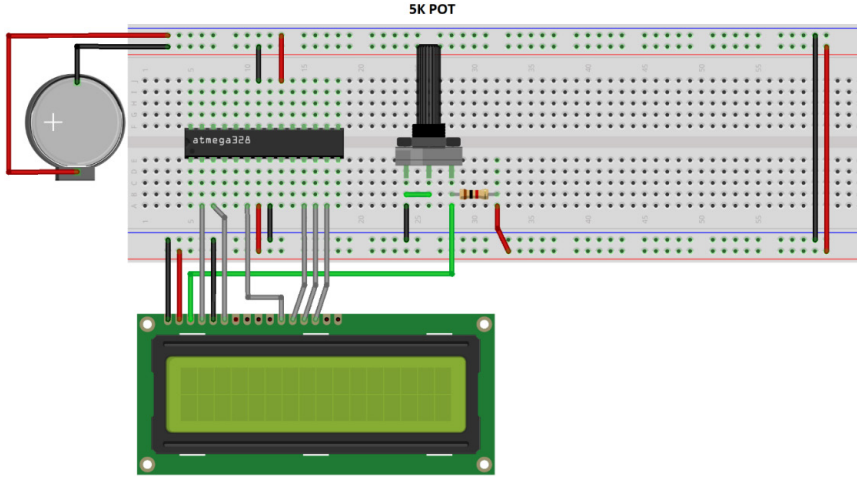
5. Adım: Devre simülasyon programında Görsel 3.27'de verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.27: 3.7 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.28’de verildiği gibi kurunuz.



Görsel 3.28: 3.7 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

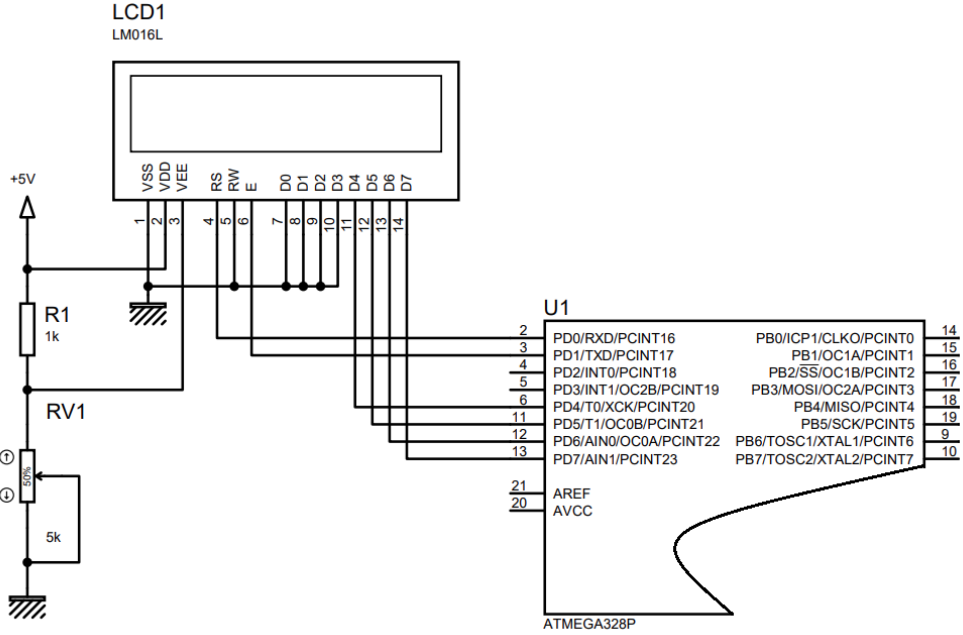
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

Mikrodenetleyiciye bağlı olan karakter LCD ekranında sınıf arkadaşlarınızın adlarını ve soyadlarını birer saniye aralıklar ile yazdıran devreyi (Görsel 3.29) iş sağlığı ve güvenliği kurallarını dikkate alarak oluşturunuz ve gömülü yazılımını kodlayınız.



Görsel 3.29: Uygulama devresi



UYGULAMA

Adı:	Karakter LCD ve Tuş Takımı	No.: 3.8
Amaç:	Mikrodenetleyici ile karakter LCD ve keypad kontrolü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda, mikrodenetleyiciye bağlı karakter LCD ekranına, tuş takımından basılan tuşun değerini yazdıran devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.11’de belirtilmiştir.

Tablo 3.11: 3.8 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
LCD	Karakter LCD (16x2)	1 adet
Keypad	3 x 4 mebran tuş takımı	1 adet
Breadboard		1 adet
Direnç	220 Ω	4 adet
Direnç	1 k Ω	1 adet
Potansiyometre	5 k Ω	1 adet
DC güç kaynağı	5 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	20 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **LCD_KEYPAD_UYGULAMASI** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#define sutun1 PORTB0 //sutun1 ifadesi PORTB0 ifadesine eşitlendi.
#define sutun2 PORTB1 //sutun2 ifadesi PORTB1 ifadesine eşitlendi.
#define sutun3 PORTB2 //sutun3 ifadesi PORTB2 ifadesine eşitlendi.
#include ".\include\lcd.h" // Projenize lcd.h dosyası dâhil edildi.
void tus_oku() //tus_oku() fonksiyonu tanımlandı.
{
    PORTB = 0b00010000; // 1. satır lojik 1 yapıldı.
    if (PINB & (1 << sutun1)) //1. sütun okunuyor.
    {
        _delay_us(20);
        LCD_Temizle(); // LCD ekranı temizlendi.
        LCD_Yaz("BASILAN TUS: 1"); // LCD'ye yaz.
    }
    if (PINB & (1 << sutun2)) //2. sütun okunuyor.
    {
        _delay_us(20);
        LCD_Temizle(); // LCD ekranı temizlendi.
        LCD_Yaz("BASILAN TUS: 2"); // LCD'ye yaz.
    }
}
```

```

if (PINB&(1<<sutun3)) //3. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 3"); // LCD'ye yaz.
}

PORTB = 0b00100000; // 2. satır lojik 1 yapıldı.
if (PINB&(1<<sutun1)) //1. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 4"); // LCD'ye yaz.
}
if (PINB&(1<<sutun2)) //2. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 5"); // LCD'ye yaz.
}
if (PINB&(1<<sutun3)) //3. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 6"); // LCD'ye yaz.
}

PORTB = 0b01000000; // 4. satır lojik 1 yapıldı.
if (PINB&(1<<sutun1)) //1. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 7"); // LCD'ye yaz.
}
if (PINB&(1<<sutun2)) //2. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 8"); // LCD'ye yaz.
}
if (PINB&(1<<sutun3)) //3. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: 9"); // LCD'ye yaz.
}

PORTB = 0b10000000; // 3. satır lojik 1 yapıldı.
if (PINB&(1<<sutun1)) //1. sütun okunuyor.
{
    _delay_us(20);
    LCD_Temizle(); // LCD ekranı temizlendi.
    LCD_Yaz("BASILAN TUS: *"); // LCD'ye yaz.
}

```

```

if (PINB&(1<<sutun2))
{
    _delay_us(20);
    LCD_Temizle();
    LCD_Yaz("BASILAN TUS: 0");
}
if (PINB&(1<<sutun3))
{
    _delay_us(20);
    LCD_Temizle();
    LCD_Yaz("BASILAN TUS: #");
}
}
int main(void)
{
    DDRB=0xF0;
    LCD_Baslat();
    while (1)
    {
        tus_oku();
        _delay_ms(150);
    }
}

```

//2. sütun okunuyor.

// LCD Ekranı Temizlendi.

// LCD'ye Yaz.

//3. sütun okunuyor.

// LCD ekranı temizlendi.

// LCD'ye yaz.

//Ana Fonskiyon

//Sütunlar giriş, satırlar çıkış yapıldı.

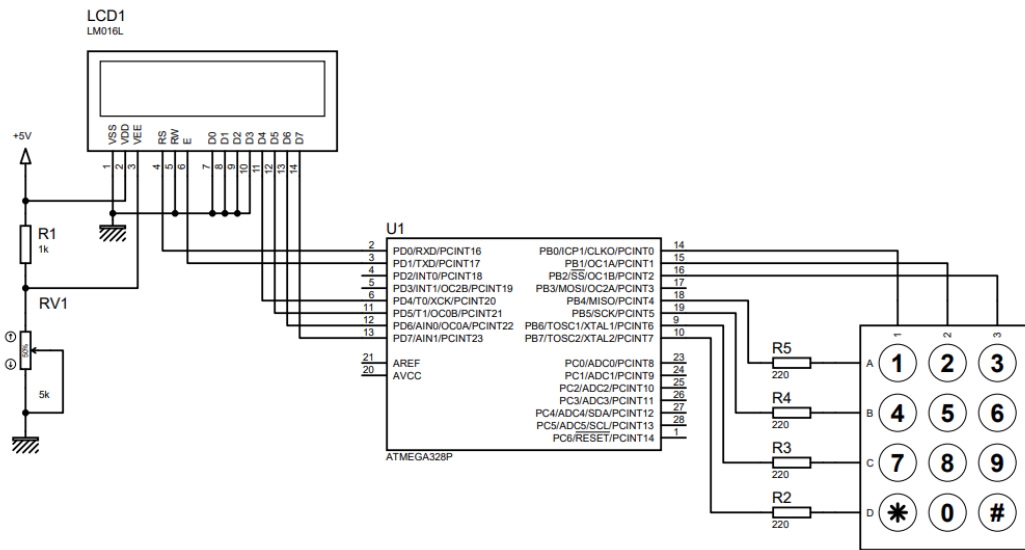
//Sonsuz döngü

//Tuş takımı okunuyor.

//Ark oluşması gecikme ile engellendi.

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

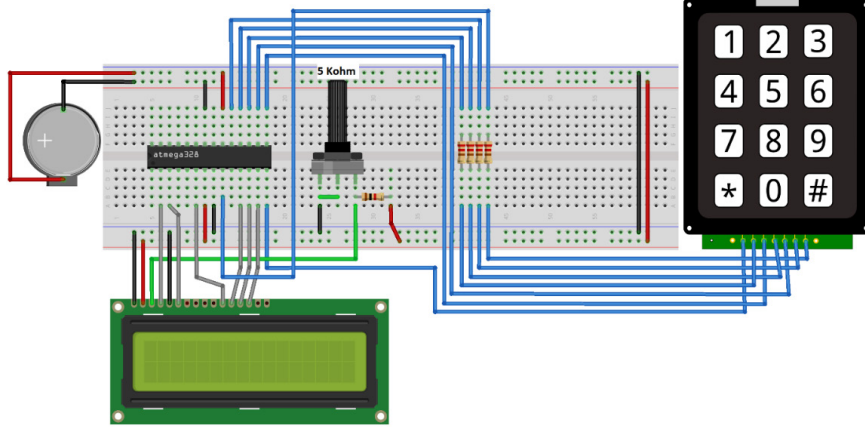
5. Adım: Devre simülasyon programında Görsel 3.30'da verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.30: 3.8 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.31’de verildiği gibi kurunuz.



Görsel 3.31: 3.7 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

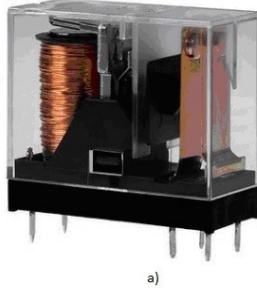
Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

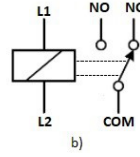
Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		

3.3. MİKRODENETLEYİCİLER RÖLE KONTROL UYGULAMALARI

Röle, düşük akım kullanarak yüksek akımlı devreleri kumanda etmeyi sağlayan elektromekanik devre elemanıdır. Görsel 3.32: a'da röle ve Görsel 3.32: b'de röle sembolü görülmektedir.

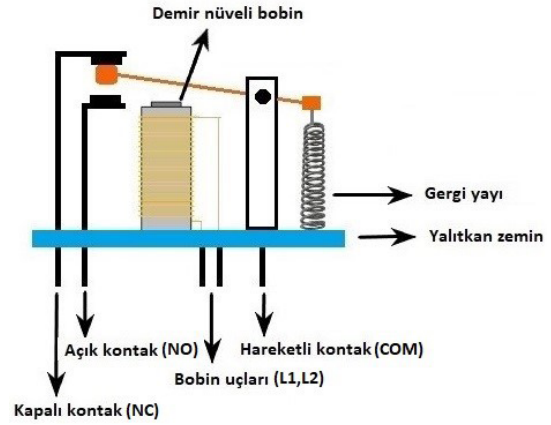


Görsel 3.32: a) Röle



Görsel 3.32: b) Röle sembolü

Röle, bobin ve kontaklardan oluşur (Görsel 3.33). Bobin ve kontaklardan oluşan röleye **manyetik röle** ismi de verilir. Bobin üzerinde enerji olmadığında hareketli kontak (COM) ile kapalı kontak (NC) birbirine bağlıdır. Bobin üzerinde enerji uygulandığında demir nüveli bobin, elektro mıknatıs özelliği kazanır ve hareketli kontağı kendine çeker. Hareketli kontak (COM) ile açık kontak (NO) birbirine bağlanır.



Görsel 3.33: Röle iç yapısı

3.3.1. Röle Çeşitleri

Manyetik röle dışında bobin, alaşım metaller veya yarı iletken teknolojisinden yararlanılarak farklı kullanım alanlarına göre çeşitli röleler üretilmektedir. En çok bilinen ve kullanılan röleler şunlardır:

- Çubuk Röle (Reed Röle)
- Termik Röle
- Katı Hâl Röle (Solid State Röle)
- Kaçak Akım Koruma Rölesi
- Zaman Rölesi
- Faz Koruma Rölesi

Manyetik röle, çubuk röle, termik röle ve katı hâl rölesi elektronik devrelerde yaygın olarak kullanılır. Diğer röleler elektrik tesisat hatlarında yaygın olarak kullanılır.

3.3.1.1. Çubuk Röle (Reed Röle)

Çubuk röle, içerişi asal gaz ile doldurulmuş ve içerisinde iki tane kontak bulunduran cam tüpten üretilen röle tipidir (Görsel 3.34).



Görsel 3.34: Çubuk röle

Çubuk rölenin kontaktları açık kontak şeklindedir. Çubuk röle üzerine bobin sarılıp, bobine enerji verilirse kontaktları kapanarak birbirine temas eder (Görsel 3.35: a). Ayrıca çubuk röle üzerine mıknatıs yaklaştırılırsa yine kontaktları kapanarak birbirine temas eder (Görsel 3.35: b).



Görsel 3.35: a) Çubuk röle bobin bağlantısı

Görsel 3.35: b) Çubuk röleye mıknatıs yaklaştırılması

3.3.1.2. Termik Röle

Termik röle, ortamdaki ısıya göre kontak bağlantıları değişen rölelerdir. Termik rölenin normalde kontağı kapalıdır. Ortamdaki ısı belirlenen ısıdan daha yüksek seviyeye ulaştığı durumlarda ise kontağı açılır. Termik rölenin kontaktları alaşım metallere (bi-metal) yapılıdır. Termik rölenin yapısındaki metallere biri fazla, diğeri daha az genişler. Genleşme farkı röledeki kontaktların açılıp kapanmasını sağlar. Görsel 3.36’da termik röle görülmektedir.



Görsel 3.36: Termik röle

3.3.1.3. Katı Hâl Rölesi (Solid State Röle)

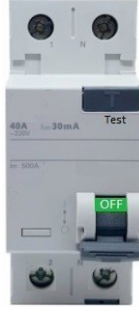
Katı hâl rölesi, yarı iletken teknolojisi kullanılarak üretilen röledir. Yapılarında mekanik parça bulunmaz. Kontaktların yerine transistör, triyak, mosfet gibi yarı iletken anahtarlama devre elemanı kullanılır. Bobin yerine ise transformatör veya opto kuplör kullanılır. AC veya DC voltajlarda kullanılabilir. Görsel 3.37’de katı hâl rölesi görülmektedir.



Görsel 3.37: Katı hâl röle

3.3.1.4. Kaçak Akım Koruma Rölesi

Kaçak akım koruma rölesi, faz ve nötr iletkenleri arasında geçen elektrik akımında fark oluştuğunda kontakları açan röle çeşididir. Elektrik devresinde fazdan giren akım ile nötrden çıkan akım normalde eşittir. Devrede kaçak akım varsa faz ile nötr arasındaki akım dengesi bozulacaktır. Kaçak akım koruma rölesi, bu dengesizliği algılayıp kontakları açar. Görsel 3.38’de bir fazlı kaçak akım koruma rölesi görülmektedir.



Görsel 3.38: Kaçak akım koruma rölesi

3.3.1.5. Zaman Rölesi

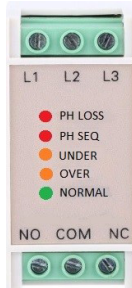
Zaman röleleri, ayarlanan süre sonunda kontakların konumunu değiştiren rölelerdir. Yapısında bulunan bobine enerji verildiğinde veya enerjisi kesildiğinde kullanıcı tarafından belirlenen süreye kadar kontak konumlarını korur. Ayarlanan süre sonunda kontaklar konum değiştirir. Zaman röleleri genellikle otomatik kumanda devrelerinde kullanılır. Görsel 3.39’da zaman rölesi görülmektedir.



Görsel 3.39: Zaman rölesi

3.3.1.6. Faz Koruma Rölesi

Faz koruma röleleri, üç fazlı elektrik motorlarını şebekede oluşabilecek hat arızalarına karşı korumak amacıyla kullanılır. Üç fazlı motorları; fazın birinin kesilmesi, faz sırasının değişmesi, fazlarda oluşacak dengesizlikler, fazlarda genlik farklılıkları veya PTC’nin ısınması durumunda genellikle kontaktör yardımıyla koruma altına alır. Görsel 3.40’ta faz koruma rölesi görülmektedir.



Görsel 3.40: Faz koruma rölesi

3.3.2. Röle Seçimi ve Bağlantısı

Elektronik devrelerde röle seçimi ve bağlantısı yapılırken dikkat edilecek hususlar şu şekildedir:

- Rölenin bobin ve kontak uçları doğru tespit edilmelidir (Genellikle röle üzerinde bobin ve kontak bacakları gösterilir). Görsel 3.41’de rölenin bobin ve kontak bağlantısı görülmektedir.



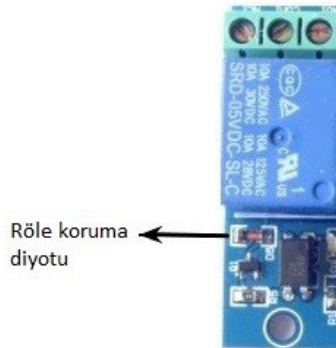
Görsel 3.41: Röle bobin ve kontak bağlantısı

- Rölenin çalışma gerilimine dikkat edilir. Rölenin besleme gerilimi röle üzerinde yazılan çalışma gerilimine uygun olmalıdır (5 volt, 12 volt, 24 volt vb.). Görsel 3.42’de rölenin çalışma gerilimi görülmektedir.



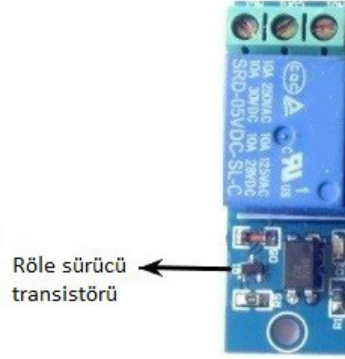
Görsel 3.42: Röle çalışma gerilimi

- Rölenin bobin uçlarına koruma diyotu takılır. Diyotun katodu, bobin besleme geriliminin artı ucuna anodu bobinin diğer ucuna bağlanmalıdır. Görsel 3.43’te röle koruma diyotu görülmektedir.



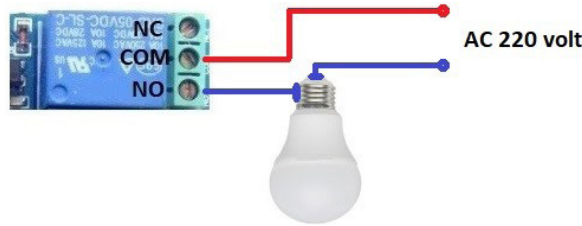
Görsel 3.43: Röle koruma diyotu

- Röle transistör veya hat sürücü entegresi ile sürülmelidir. Görsel 3.44'te röle sürücü transistörü görülmektedir.



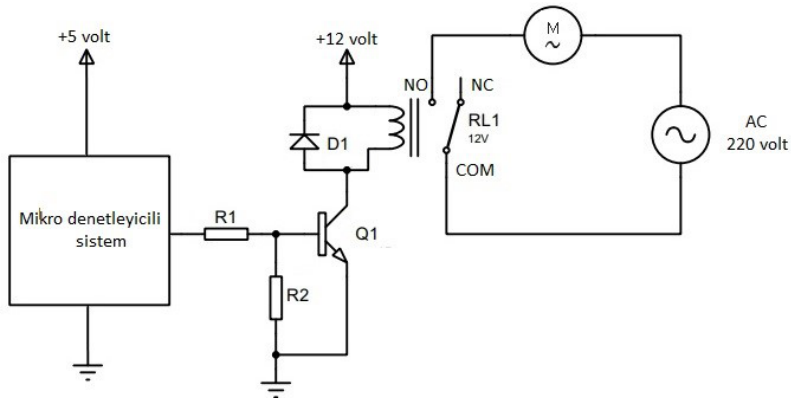
Görsel 3.44: Röle sürücü transistörü

- Röleye bağlanacak yük devresi her zaman kontak uçlarına bağlanmalıdır. Görsel 3.45'te röle yük bağlantısı görülmektedir.

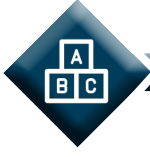


Görsel 3.45: Röle yük bağlantısı

Röle çoğunlukla elektronik devre veya mikro denetleyicili sistemler tarafından kontrol edilir. Röle besleme gerilimi her zaman elektronik devrenin besleme gerilimi ile aynı olmayabilir. Görsel 3.46'da 5 voltta çalışan mikro denetleyicili sistem ile 12 voltluk röle sürülmüştür. Röle kontaklarına ise 220 voltta çalışan AC motor bağlanmıştır.



Görsel 3.46: Yüksek akımlı bir devrenin röle ile kontrolü



UYGULAMA

Adı:	Mikrodenetleyici ile röle sürmek	No.: 3.9
Amaç:	Mikrodenetleyici ile röle kontrolünü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, devreye enerji uygulandıktan beş saniye sonra, PD0 pinine bağlı röleyi çalıştıran devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.12’de belirtilmiştir.

Tablo 3.12: 3.9 No.lu Uygulama İçin Malzeme Listesi

Adı	Özellği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Röle	5 volt	5 adet
Direnç	1 kΩ	5 adet
Direnç	10 kΩ	1 adet
Kondansatör	100 nF	1 adet
Diyot	1n4007	1 adet
LED diyot	Kırmızı	1 adet
Transistör	BC337	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Atmel Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 3.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **role_uyg** yazınız.

2. Adım: Görsel 3.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL           //İşlemci hızı belirlenir.
#include <avr/io.h>                //Giriş çıkış port ayar dosyası programa dâhil edilir.
#include <util/delay.h>           //Zaman geciktirme dosyası programa dâhil edilir.

int main(void)
{
    DDRD = 0xFF;                 //PORTD’nin tamamı çıkış olarak ayarlanır.
```

```

PORTD=0x00;                                     //PORTD'nin tamamına lojik 0 bilgisi verilerek sıfırlanır.

while(1){

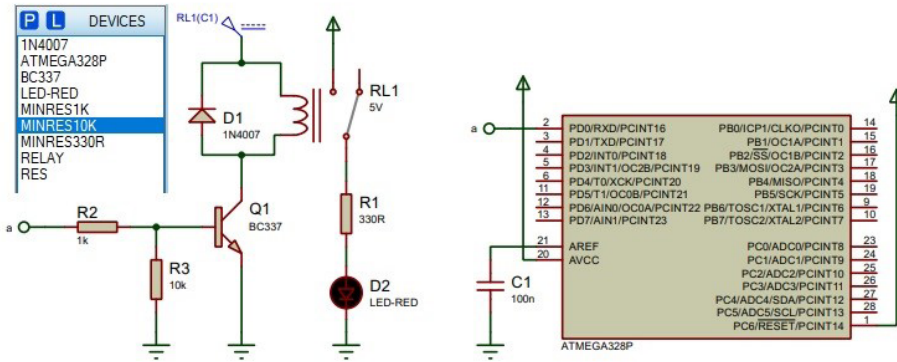
    _delay_ms(1000);                             //1 saniye gecikme yapılır.
    _delay_ms(1000);                             //1 saniye gecikme yapılır.
    _delay_ms(1000);                             //1 saniye gecikme yapılır.
    _delay_ms(1000);                             //1 saniye gecikme yapılır.
    _delay_ms(1000);                             //1 saniye gecikme yapılır.
    PORTD = 0x01;                                //D portunun sıfırncı pini lojik 1 (5 volt) yapılır.

}
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

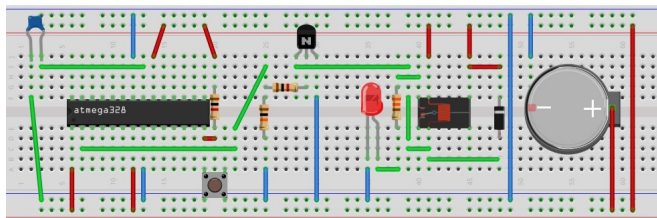
5. Adım: Devre simülasyon programında Görsel 3.47'de verilen devreyi kurarak simüle ediniz.



Görsel 3.47: 3.9 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.48'de verildiği gibi kurunuz (Elinizde bulunan röleye göre bobin ve kontak bağlantılarını devreyi dikkate alarak bağlayınız.) ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 3.48: 3.9 No.lu uygulama için breadboard üzerine kurulan devre

8. Adım: Devreye enerji vererek beş saniye bekleyiniz. LED diyotun ışık verip vermediğini kontrol ediniz.

9. Adım: Devrenin çalışmadıysa 7 ve 8. adımları tekrarlayınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi doğru olarak gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Ayarlanan süre sonunda rölenin çalışması izlendi.		
6. Temizliğe dikkat edildi.		



UYGULAMA

Adı:	Mikrodenetleyici ve Buton ile Röle Sürmek	No.: 3.10
Amaç:	Mikrodenetleyici ve buton ile röle kontrolünü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda; mikro denetleyicinin PBO pinine bağlı butona basıldığında PDO pinine bağlı röleyi çalıştıran, butona tekrar basıldığında röleyi durduran devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.13'te belirtilmiştir.

Tablo 3.13: 3.10 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Röle	5 volt	5 adet
Direnç	1 kΩ	5 adet
Direnç	10 kΩ	1 adet
Kondansatör	100 nF	1 adet
Diyot	1n4007	1 adet
LED diyot	Kırmızı	1 adet
Transistör	BC337	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Atmel Studio’da **File** menüsünden **New Project** seçeneğini seçiniz. Görsel 2.62’de görülen şekilde, **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **buton_role_uyg** yazınız.

2. Adım: Görsel 3.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL //İşlemci hızı belirlenir.
#include <avr/io.h> //Giriş çıkış port ayar dosyası programa dâhil edilir.
#include <util/delay.h> //Zaman geciktirme dosyası programa dâhil edilir.

#define ROLE 0 // ROLE ifadesinin karşılığına 0 değeri tanımlanır.

int main (void)
{
    DDRD=0xFFu; //PORTD’nin tamamı çıkış olarak ayarlanır.
    DDRB &= ~(1<<0); //PORTB’nin sıfırncı pini giriş olarak ayarlanır.
    PORTB = 0x00; //PORTB’nin tamamına lojik 0 bilgisi verilerek sıfırlanır.
    PORTD = 0x00; //PORTD’nin tamamına lojik 0 bilgisi verilerek sıfırlanır.

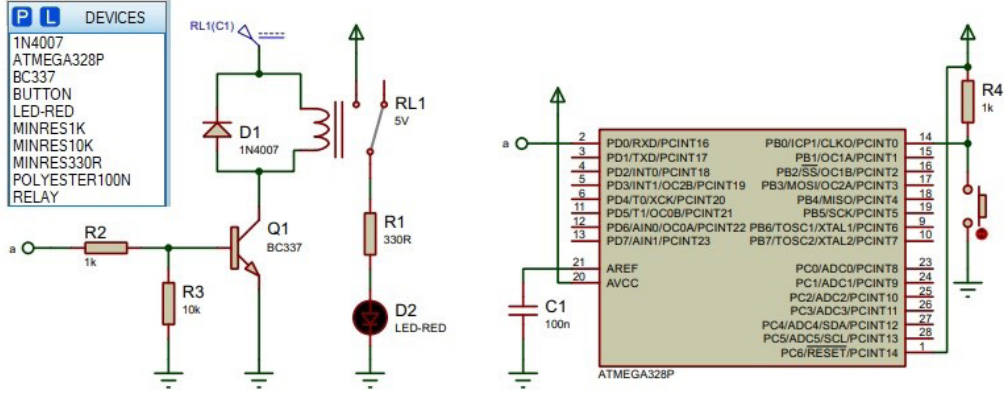
    while(1){

        if (!(PINB & (1<<0))) //Butona basıldıysa parantez içindeki işlemler yapılır.
        {
            PORTD ^= (1<<ROLE); //PORTD’nin sıfırncı pinindeki lojik değer değiştirilir.
            while (!(PINB & (1<<0))); //Buton bırakılıncaya kadar beklenir.
        }

    }
}
```


4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz.

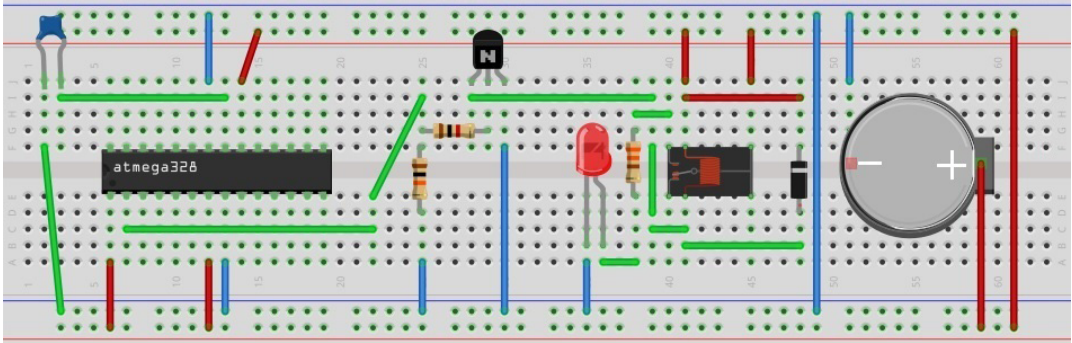
5. Adım: Devre simülasyon programında, Görsel 3.49'da verilen devreyi kurarak simüle ediniz.



Görsel 3.49: 3.10 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.50'de verildiği gibi kurunuz (Elinizde bulunan röleye göre bobin ve kontak bağlantılarını, devreyi dikkate alarak bağlayınız.) ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 3.50: 3.10 No.lu uygulama için breadboard üzerine kurulan devre

8. Adım: Devreye enerji vererek butona basınız. Rölenin çalışıp LED diyotun ışık verdiğini gözlemleyiniz. Butona tekrar basınız. Rölenin durup LED diyotun sönüğünü gözlemleyiniz.

9. Adım: Devrenin çalışmadıysa 7 ve 8. adımları tekrarlayınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

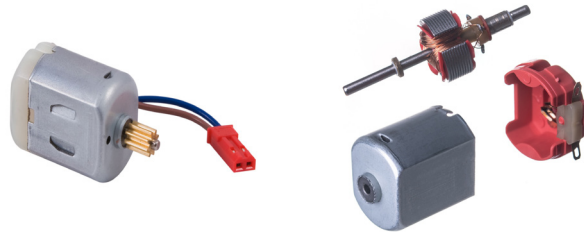
Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi doğru olarak gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Butona basıldığında rölenin çalışıp LED diyotun ışık verdiği görüldü.		
6. Butona tekrar basıldığında rölenin durup LED diyotun söndüğü görüldü.		
7. Temizliğe dikkat edildi.		

3.4. MİKRODENETLEYİCİ İLE MOTOR KONTROL UYGULAMALARI

Bu konu başlığı altında DC motor ve step (adım) motorlar hakkında bilgi verilecek ve mikrodenetleyiciler ile motorların kontrolleri gerçekleştirilecektir.

3.4.1. DC Motor

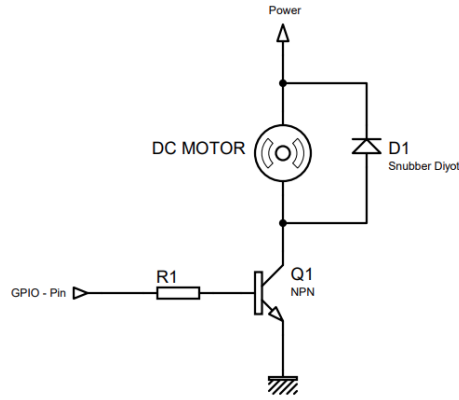
DC motorlar; içinden akım geçen bir iletkenin, manyetik alan içine yerleştirilmesiyle dönme hareketinin elde edilmesi prensibine göre çalışan makinelerdir. DC motorlar bünyesinde mıknatıs ve bir adet rotor (pervane) bulundurur. Görsel 3.51’de bir DC motorun iç yapısı ve dış görünüşü verilmiştir.



Görsel 3.51: DC motorun dış görünüşü ve iç yapısı

DC motoru, mikrodenetleyicinin pinine direkt ya da doğrudan bağlamak sağlıklı bir işlem değildir. Bunun sebebi denetleyicinin pinleri motoru sürmek için gerekli olan yeterli akımı sağlayamaz. Ayrıca motorlar elektromıknatıs sargılara sahip olduklarından indüktör gibi davranarak **ters EMF (Electromotive Force)** oluşturur. Ters EMF, devrenin bir koruması yoksa denetleyiciye zarar verebilir.

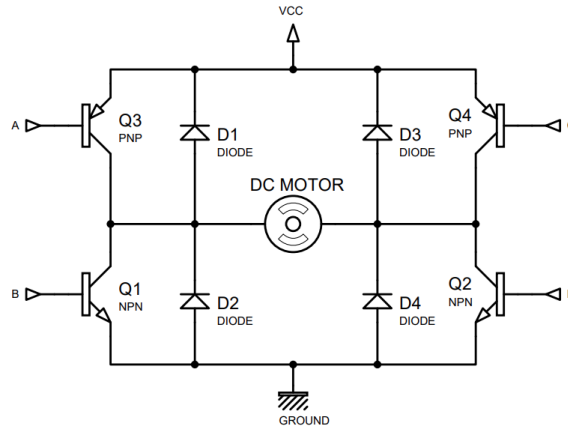
Motorların enerjisi kesildiğinde anlık olarak duramaz. Bir süre dönmeye devam eder. Elektromıknatıs sargılara sahip olan motorlarda bu durum, motoru kısa süre gerilim üreten jeneratöre dönüştürür. Mikrodenetleyiciye bağlanan motor ile denetleyici arasında bir koruma devresi bulunmaz ise bu gerilim darbeleri denetleyiciye zarar verir. Bu olasılığın önüne geçmek için motora paralel olarak bir diyot bağlanır. Bu diyota **snubber (sönümlleme) diyotu** denir. Görsel 3.52’de snubber diyotun bağlantı şeması verilmiştir.



Görsel 3.52: Snubber Diyot Bağlantısı

3.4.1.1. Transistörlü H-Köprü Sürücü Devresi

DC motorların kontrolünde H-Köprü devreleri kullanılır. Bu devreler transistörlerle oluşturulur. Bu devre yardımı ile motorun ileri ve geri yönlü döndürme hareketini sağlanabilir. Görsel 3.53’te transistörlü H-Köprü sürücü devresi verilmiştir.



Görsel 3.53: Transistörlü H-Köprüsü sürücü devresi

Tablo 3.14’te, Görsel 3.53’te verilen transistörlü H-Köprü sürücü devresindeki girişlere uygulanacak sinyallere göre motorun durumu verilmiştir.

Tablo 3.14: H-Köprü Sürücü Devresi Giriş Sinyallerine Göre Motorun Durumu

A	B	C	D	Motor Durumu
1	1	0	0	Motor saat yönünde döner.
0	0	1	1	Motor saat yönünün tersine döner.
1	0	1	0	Motor durur (OFF).
0	0	0	0	Motor boşta.



UYGULAMA

Adı:	DC Motor Uygulaması 1	No.: 3.11
Amaç:	Mikrodenetleyici ile DC motor kontrolünü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda mikrodenetleyicinin; PD5 pinine bağlı olan butona basıldığında, B portuna bağlı motoru saat yönünde döndüren, PD6 pinine bağlı olan butona basıldığında B portuna bağlı motoru durduran ve PD7 pinine bağlı olan butona basıldığında B portuna bağlı motoru saat yönünün tersi yönde döndüren devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.15’te belirtilmiştir.

Tablo 3.15: 3.10 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Buton		3 adet
Direnç	220 Ω	7 adet
Direnç	10 k Ω	3 adet
Kondansatör	100 nF	1 adet
PNP transistör	BC 212 veya BC 213	2 adet
NPN transistör	BC 237 veya BC 547	2 adet
Diyot	1N4007	4 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
DC motor	5 V – 12 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	30 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **DC_MOTOR_UYGULAMASI_1** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

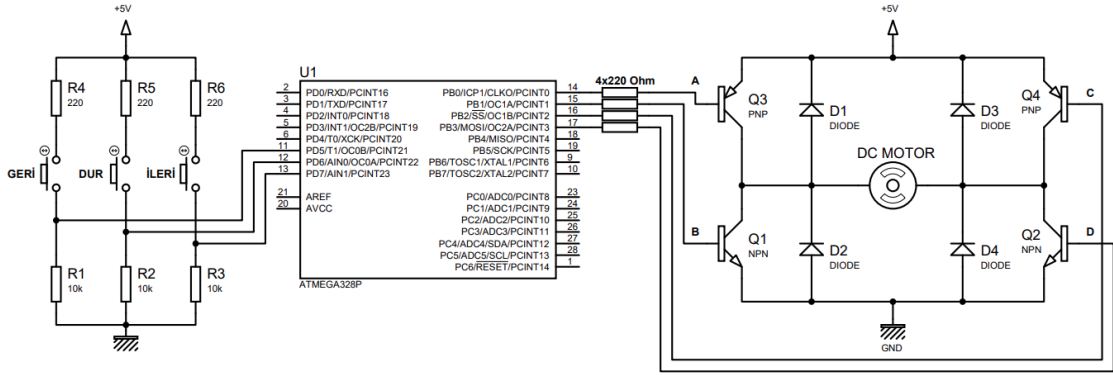
3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#define ileri PORTD7 //ileri ifadesi PORTD7 ifadesine eşitlendi.
#define dur PORTD6 //dur ifadesi PORTD6 ifadesine eşitlendi.
#define geri PORTD5 //geri ifadesi PORTD5 ifadesine eşitlendi.

int main(void)
{
    DDRB = 0xFF; // B portu çıkış olarak ayarlandı.
    DDRD = 0x00; // D portu giriş olarak ayarlandı.
    while (1)
    {
        if(PIND & (1<<ileri)) //ileri butonuna basıldıysa yapılacaklar:
        {
            _delay_ms(15);
            PORTB = 0b00000011; // A-B girişleri lojik 1 yapıldı (Motor, saat yönünde döner.).
        }
        if(PIND & (1<<dur)) //dur butonuna basıldıysa yapılacaklar
        {
            _delay_ms(15);
            PORTB = 0b00000101; // A-C girişleri lojik 1 yapıldı (Motor durur.).
        }
        if(PIND & (1<<geri)) //geri butonuna basıldıysa yapılacaklar
        {
            _delay_ms(15);
            PORTB = 0b00001100; // C-D girişleri lojik 1 yapıldı (Motor saat yönünün tersine döner.).
        }
    }
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

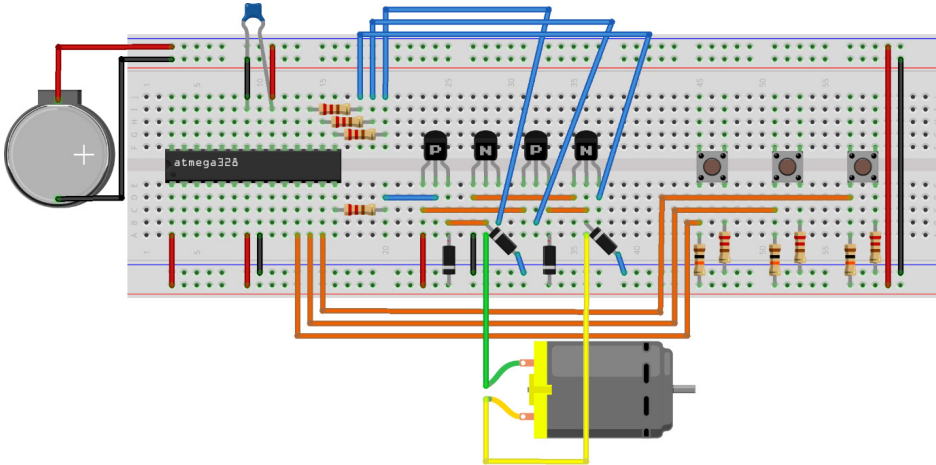
5. Adım: Devre simülasyon programında Görsel 3.54’te verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.54: 3.11 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.55'te verildiği gibi kurunuz.



Görsel 3.55: 3.11 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		

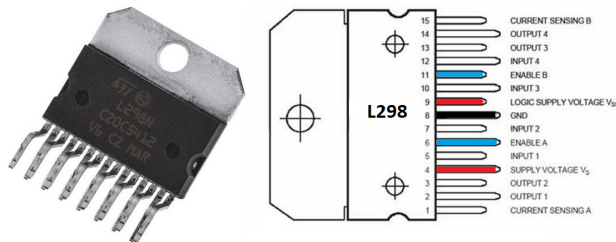
3.4.1.2. DC Motor Sürücü Entegreleri

Piyasada içerisinde H-Köprü sürücü devresi barındıran birçok entegre bulunmaktadır. L293, L298, L6201, L6202, L6203 ve LMD18200 entegreleri içerisinde H-Köprü sürücü devresi barındıran DC motor sürücü entegreleridir.

Bu öğrenme biriminde L298 DC motor sürücü entegresi ile motor kontrolü incelenecektir.

L298 DC motor sürücü entegresinde 2 adet transistörlü H köprü sürücü devresi vardır. Bu devreler A ve B olarak isimlendirilmektedir. Bu entegre ile iki adet motor kontrolü yapılabilir. L298 entegresinin çıkışı akımı 4 amper kadar ulaşabilmektedir. Entegre 46 V'a kadar çalışma gerilimi ile beslenebilir. Ayrıca entegre aşırı ısı korumasına da sahiptir.

Görsel 3.56'da, L298 entegresinin dış görünümü ve pin isimleri verilmiştir.



Görsel 3.56: L298 entegresinin dış görünümü ve pin isimleri

Tablo 3.16'da, L298 entegresinin pinlerinin görevleri verilmiştir.

Tablo 3.16: L298 Entegre Pinlerinin Görevleri

Pin No.	Pin İsmi	Açıklama
1, 15	Current Sensing A-B	Bu pinler toprak (GND) hattına bir direnç ile bağlanır. Bu pinler motordan geçen akımı kontrol etmek için kullanılan pinlerdir.
2, 3	Output 1-2	A köprüsünün çıkış uçlarıdır.
4	Supply Voltage VS	Motorun pozitif besleme ucudur. Bu uç 100 nF'lik dekuplaj kondansatörü ile şase ucuna bağlanır.
5, 7	Input 1-2	A köprüsünün kontrol uçlarıdır.
6, 11	Enable A-B	A ve B köprülerini aktif etme pinleridir.
8	GND	Toprak (Şase) ucudur.
9	VSS	Entegrenin besleme ucudur. Bu uç 100 nF'lik dekuplaj kondansatörü ile şase ucuna bağlanır.
10, 12	Input 3-4	B köprüsünün kontrol uçlarıdır.
13, 14	Output 3-4	B köprüsünün çıkış uçlarıdır.

L298 entegresinin girişlerine uygulanan lojik değerlerine göre motorun alacağı hareket durumları Tablo 3.17'de verilmiştir.

Tablo 3.17: L298 Girişlerine Göre Motor Çalışma Durumları

Köprü	Girişler	Durum
A Köprüsü	Enable A = HIGH Enable B = LOW Input1 = HIGH Input2 = LOW	İleri
	Enable A = HIGH Enable B = LOW Input1 = LOW Input2 = HIGH	Geri
	Enable A = HIGH Enable B = LOW Input1 = Input2 (Input1 ve Input2 aynı seviyede olmalı)	Hızlı stop
	Enable A = HIGH Enable B = LOW Input1 = X Input2 = X	Motor boşta
B Köprüsü	Enable A = LOW Enable B = HIGH Input1 = HIGH Input2 = LOW	İleri
	Enable A = LOW Enable B = HIGH Input1 = LOW Input2 = HIGH	Geri
	Enable A = LOW Enable B = HIGH Input1 = Input2 (Input1 ve Input2 aynı seviyede olmalı)	Hızlı stop
	Enable A = LOW Enable B = HIGH Input1 = X Input2 = X	Motor boşta



UYGULAMA

Adı:	DC Motor Uygulaması- 2	No.: 3.12
Amaç:	L298 entegresi kullanılarak motorun yön kontrolünü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, mikrodenetleyicinin PD2 pinine bağlı butona basıldığında B portuna bağlı motoru, saat yönünde döndüren; PD1 pinine bağlı butona basıldığında B portuna bağlı motoru, durduran ve PD0 pinine bağlı butona basıldığında B portuna bağlı motoru, saat yönünün tersi yönde döndüren; L298 entegresi ile kurulmuş devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.18’de belirtilmiştir.

Tablo 3.18: 3.12 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Buton		3 adet
Direnç	220 Ω	3 adet
Direnç	10 k Ω	3 adet
Direnç	100 Ω	2 adet
Kondansatör	100 nF	2 adet
Pil	9 V	1 adet
Diyot	1N4007	4 adet
Entegre	L298	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
DC motor	5 V-12 V	1 adet
Bağlantı kabloları	Çeşitli renklerde	30 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **DC_MOTOR_UYGULAMASI_2** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: main.c dosyası içerisine aşağıda verilen kodları yazınız.

```

#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#define ileri PORTD2
#define dur PORTD1
#define geri PORTD0

int main(void)
{
    DDRB = 0xFF;
    DDRD = 0x00;

    while (1)
    {
        if(PIND & (1<<ileri))
        {
            _delay_ms(15);
            PORTB = 0b00000101;
        }
        if(PIND & (1<<dur))
        {
            _delay_ms(15);
            PORTB = 0b00000100;
        }
        if(PIND & (1<<geri))
        {
            _delay_ms(15);
            PORTB = 0b00000110;
        }
    }
}

```

//ileri ifadesi PORTD2 ifadesine eşitlendi.
//dur ifadesi PORTD1 ifadesine eşitlendi.
//geri ifadesi PORTD0 ifadesine eşitlendi.

// B portu çıkış olarak ayarlandı.
// D portu giriş olarak ayarlandı.

// ileri butonuna basıldıysa yapılacaklar:

// A köprüsü aktif. Input - 1 = HIGH, Input - 2= LOW

// Dur butonuna basıldıysa yapılacaklar:

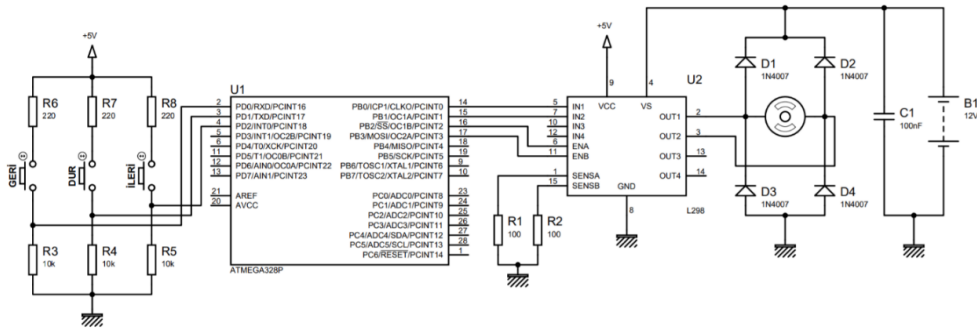
// A köprüsü aktif. Input - 1 = Input - 2 = LOW

// Geri butonuna basıldıysa yapılacaklar:

// A köprüsü aktif. Input - 1 = LOW, Input - 2= HIGH

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

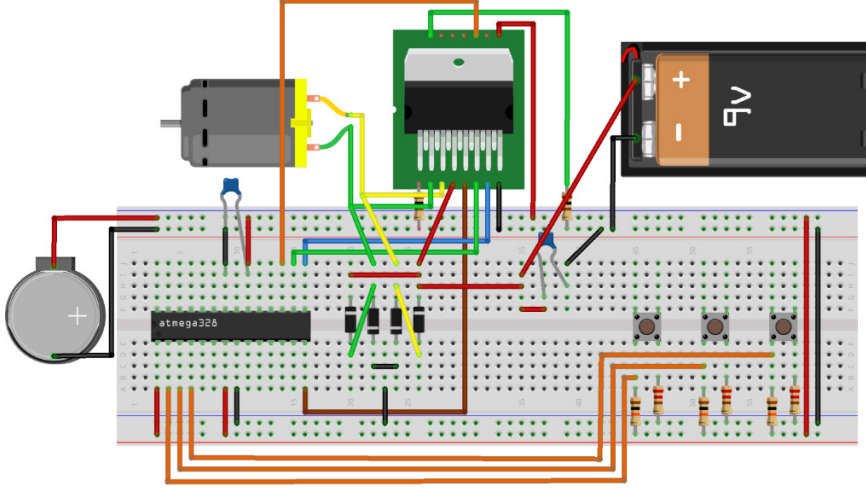
5. Adım: Devre simülasyon programında Görsel 3.57’de verilen devreyi kurunuz ve simüle ediniz.



Görsel 3.57: 3.12 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.58’de verildiği gibi kurunuz.



Görsel 3.58: 3.12 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

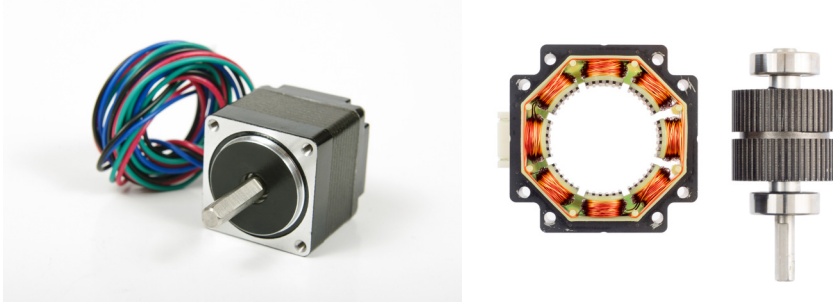
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		

3.4.2. Step Motorlar

Step motorlar; yapısında dijital sinyalle kontrol edilebilen bobinler (stator) bulunduran ve adım adım hareket edebilen motorlardır. Dijital sinyal yardımı ile belirli açıda dönme hareketi sağlanabilir. Bu özellikleri sayesinde step motorlar, bilgisayar destekli uygulamalarda (yazıcılar, hard diskler, robotik alanı, konum kontrol sistemleri vb.) çok fazla tercih edilirler.

Step motorlar; rotor, stator ve rulmandan meydana gelir. Görsel 3.59’da step motorun dış görünüşü ve iç yapısı görselleştirilmiştir.



Görsel 3.59: Step Motor dış ve iç görünüşü

Step motorun hareketsiz olan ve sargılardan oluşan kısmına **stator** adı verilir. **Rotor** ise motorun hareketli ve N, S kutbundan meydana gelen ve tek parça sabit mıknatıstan oluşan kısımdır. **Rulman;** motorun hareketini en az sürtünme ile sağlamak ve oluşan torktan (güç) en az kayıp vererek motorun hareketini sağlamak için kullanılan kısımdır.

Stator üzerindeki her bobin birbirinden bağımsız olarak dijital olarak aktif edilebilmektedir. Stator üzerinde bulunan bobin sayısı step motorların hepsinde aynı sayıda değildir. Fakat genel olarak sekiz adet bulunmaktadır. Stator üzerindeki bobin sayısı arttıkça daha hassas hareket elde edilebilir.

Step motorun çalışma prensibi; rotor üzerinde N, S kutuplarına sahip bir mıknatıs mevcuttur. Stator üzerindeki bobinler enerjilendirildiğinde manyetik alan oluşturmaktadır. Rotor oluşan bu manyetik alandan etkilenecek enerjilenen bobinin karşısına gelir ve durur. Stator üzerindeki bobinler sırası ile enerjilendirilerek motorun dönme hareketi oluşturulmaktadır. Stator uçlarına uygulanan akımın yönü değiştirilerek motorun dönüş yönü kontrol edilebilir.

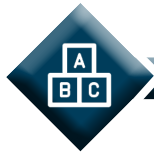
Step motor kontrol edilirken mikrodenetleyicinin dijital pinlerinden lojik 1 ve lojik 0 değerleri gönderilerek stator üzerinde yer alan bobinler aktif edilir. Atmega328P denetleyicisinin dijital giriş / çıkış pinleri 40 mA akım sağlamaktadır. Bu akım step motoru sürmek için yeterli bir değer değildir. Bunun için denetleyicinin dijital çıkış pinlerinden uygulanan akımın yükseltilmesi gerekmektedir. Bunun için hazır entegreler kullanılmaktadır. Piyasada step motor sürmek için ULN2003 entegresi çok sık kullanılmaktadır. ULN2003 entegresi girişinden uygulanan sinyali 500 mA’ye kadar yükseltebilmektedir. Ayrıca çıkışından 50 V’a kadar çıkış sağlayabilmektedir. Bu değer step motor sürmek için yeterli bir değerdir.

Step motorlar piyasada 4, 5 ve 6 uçlu (kablolu) olarak bulunmaktadır. Bu öğrenme biriminde 6 uçlu step motor kullanılacaktır. 6 uçlu step motorlarda 2 uç ortak uç kullanılmaktadır. Ortak uçlar isteğe bağlı + besleme ya da toprak hattı olarak kullanılabilir. Diğer 4 uç ise stator bobin uçlarıdır.

Step motorlar sürülürken “**tam adım**” ve “**yarım adım**” adı verilen sürme teknikleri kullanılır. Tam adım sürme tekniğinde statorda bulunan bobinlerden karşılıklı olan iki bobine aynı anda enerji verilir. Bu sayede yaklaşık 1,5 kat daha fazla tork elde edilir. Fakat çekilen akım iki katına çıkmaktadır. Yarım adım sürme tekniğinde ise tek bobin enerjilendirildikten sonra sıradaki bobin ile aktif olan bobin aynı anda enerjilendirilir. Böylece rotor yarım adım döndürülmüş olur. Daha hassas hareket gerektiren uygulamalarda tercih edilen bir yöntemdir. Tablo 3.19’da tam adım ve yarım adım sürme teknikleri için uçlara verilmesi gereken sinyallerin değerleri verilmiştir.

Tablo 3.19: Tam ve Yarım Adım Sürme

Adım	Tam Adım					Yarım Adım				
	Uç-1	Uç-2	Uç-3	Uç-4	HEX	Uç-1	Uç-2	Uç-3	Uç-4	HEX
1	1	0	0	1	0x09	1	0	0	0	0x01
2	1	1	0	0	0x03	1	1	0	0	0x03
3	0	1	1	0	0x06	0	1	0	0	0x02
4	0	0	1	1	0x09	0	1	1	0	0x06
5						0	0	1	0	0x04
6						0	0	1	1	0x0C
7						0	0	0	1	0x08
8						1	0	0	1	0x09



UYGULAMA

Adı:	Step motor Uygulaması- 1	No.: 3.13
Amaç:	ULN2003 entegresi kullanılarak step motorun yön kontrolünü yapmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda, mikrodenetleyicinin PD0 pinine bağlı butona basıldığında B portuna bağlı motoru, saat yönünde döndüren; PD1 pinine bağlı butona basıldığında B portuna bağlı motoru, saat yönünün tersi yönde döndüren; ULN2003 entegresi ile kurulmuş devreyi oluşturunuz ve gömülü yazılımını kodlayınız.

Kullanılacak Araç Gereç: Tablo 3.20’de belirtilmiştir.

Tablo 3.20: 3.13 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328P	1 adet
Buton		2 adet
Direnç	220 Ω	2 adet
Direnç	10 k Ω	2 adet
Kondansatör	100 nF	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Step motor	6 uçlu	1 adet
Bağlantı kabloları	Çeşitli renklerde	25 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet

Uygulama Adımları:

1. Adım: Microchip Studio’da “File” menüsünden “New Project” seçeneğini seçiniz. Görsel 2.62’de görülen “GCC C Executable Project” seçeneğini seçtikten sonra “Name” kısmına **STEP_MOTOR_UYGULAMASI_1** yazınız.

2. Adım: Görsel 2.63’te görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 1000000UL //Gecikme fonksiyonu için kullanılacak osilatör frekansı
                          //belirleniyor.

#include <avr/io.h>
#include <util/delay.h>
#define ileri PORTD1 //İleri ifadesi PORTD1 ifadesine eşitlendi.
#define geri PORTD0 //Geri ifadesi PORTD0 ifadesine eşitlendi.

// Step motor için yarım adım dönüş bilgileri tanımlanıyor.
const int adimlar[] = {0x01,0x03,0x02,0x06,0x04,0x0C,0x08,0x09};
int main(void)
{
    int adim=0;
    int hiz=400;
    DDRB = 0xFF; //B portu çıkış olarak ayarlandı.
    DDRD = 0x00; //D portu giriş olarak ayarlandı.
    while (1)
    {
        if(PIND&(1<<ileri)) // İleri butonuna basıldıysa yapılacaklar:
        {
            PORTB = adimlar[adim]; // Sıradaki yarım adım değerini B portuna
                                  //gönder.
            while(PIND&(1<<ileri)) //Butonu bırakana kadar bekle.
                _delay_ms(hiz);
        }
    }
}
```

```

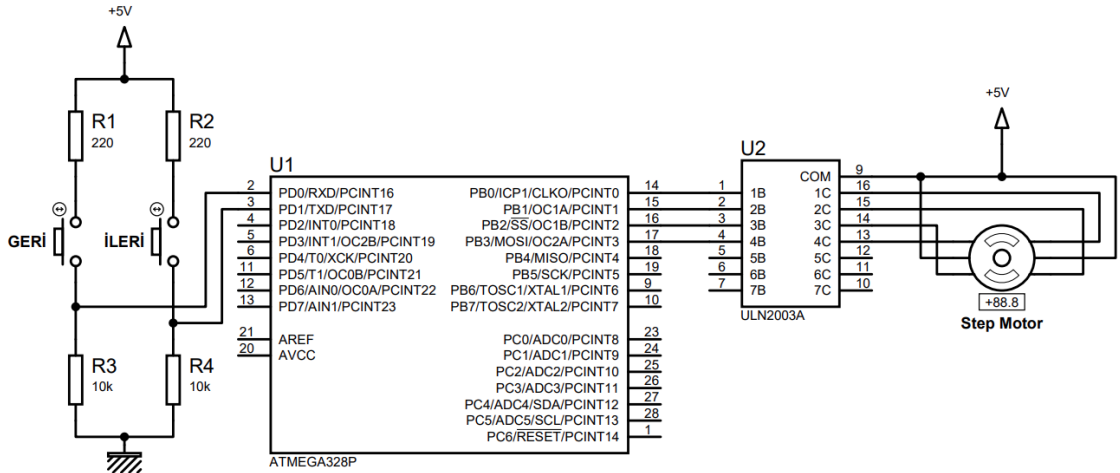
        if(adim==7)                                //adim değişkeni 7 olduğunda adim
                                                    değişkeninin değeri -1 olsun.
            adim=-1;
        adim++;                                    //adim değişkeninin değerini 1 arttır.
    }

    if(PIND&(1<<geri))                            // Geri butonuna basıldıysa yapılacaklar:
    {
        if(adim==0)                                //adim değişkeni 0 olduğunda adim
                                                    değişkeninin değeri 8 olsun.
            adim=8;
        adim--;                                    //adim değişkeninin değerini 1 azalt.
        PORTB = adimlar[adim];                    // Sıradaki yarım adım değerini B portuna
                                                    gönder.
        while(PIND&(1<<geri))                      //Butonu bırakana kadar bekle.
            _delay_ms(hiz);
    }
}
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

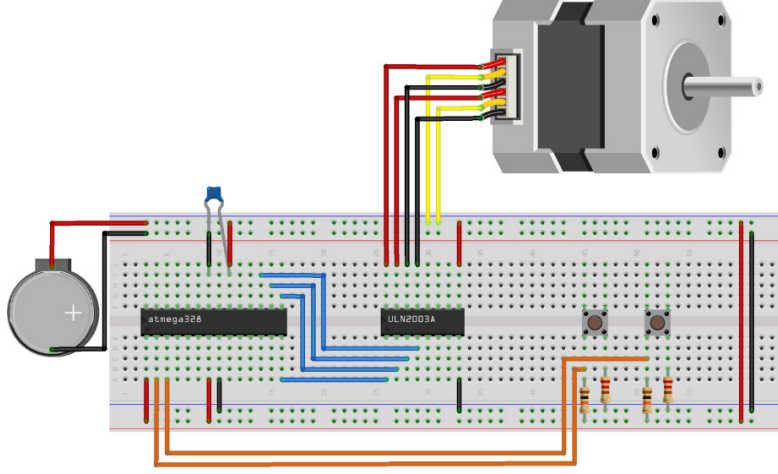
5. Adım: Devre simülasyon programında GörSEL 3.60'ta verilen devreyi kurunuz ve simüle ediniz.



GörSEL 3.60: 3.13 No.lu uygulama için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine, Görsel 3.61’de verildiği gibi kurunuz.



Görsel 3.61: 3.7 No.lu uygulama için breadboard eleman dizilimi

8. Adım: Mikrodenetleyici programlayıcısı yardımı ile denetleyicinizi programlayınız.

9. Adım: Öğretmeninizden onay aldıktan sonra devreye enerji veriniz ve çalıştırınız.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Devreye enerji verildiğinde başarılı bir şekilde devre çalıştı.		
6. Temizliğe dikkat edildi.		

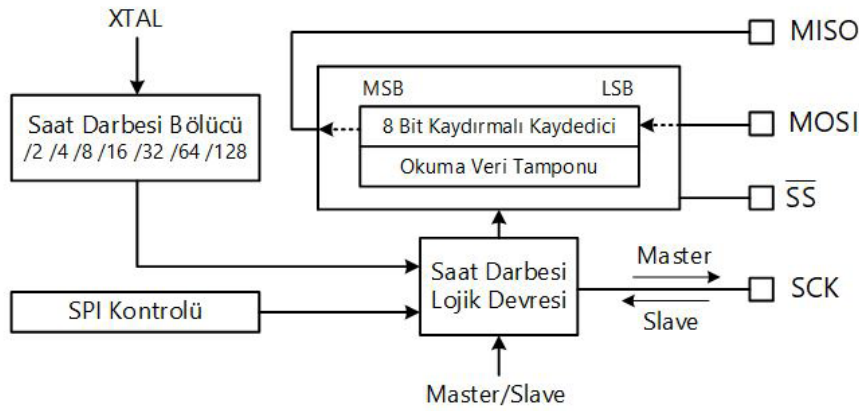
3.5. MİKRODENETLEYİCİ İLE HABERLEŞME UYGULAMALARI

Mikrodenetleyicinin diğer mikrodenetleyiciler ve çevre cihazlar ile haberleşmesi için farklı yöntemler bulunmaktadır. Bu yöntemlerden en sık kullanılanları aşağıda verilmiştir.

- Seri çevresel arayüz (SPI)
- Evrensel senkron ve asenkron seri verici ve alıcı (USART)
- İki hatlı seri arayüz (TWI)

3.5.1. SPI Haberleşmesi

Seri çevresel arayüz (SPI), mikrodenetleyici ve çevre cihazlar arasında yüksek hızda senkron (eş zamanlı) veri aktarımına imkân sağlayan bir haberleşme yöntemidir. Atmega328P mikrodenetleyici içerisinde SPI haberleşme modülü yer alır. SPI haberleşme modülünün basitleştirilmiş blok şeması Görsel 3.62’de verilmiştir.



Görsel 3.62: SPI haberleşme modülü basitleştirilmiş blok şeması

SPI modülünde 8 bitlik kaydırmalı SPI veri kaydedicisi (SPDR) bulunur. Kaydırmalı kaydedici, karşı cihazdan gelen bitleri varsayılan olarak en düşük değerli bittten [Least Significant Bit (LSB)] en yüksek değerli bite [Most Significant Bit (MSB)] doğru birer birer kaydırarak kaydeder ve okuma işlemini gerçekleştirir. Bu işlem sırasında, alınan bitler öncelikle okuma veri tamponuna, okuma işlemi tamamlandıktan sonra SPI veri kaydedicisine yazılır.

Yazma işlemi yapılırken ise SPI veri kaydedicisindeki bitler varsayılan olarak MSB’den başlanıp LSB’ye doğru birer birer kaydırılarak karşı cihaza gönderilir. Yazma işlemi esnasında okuma veri tamponu kullanılmaz.

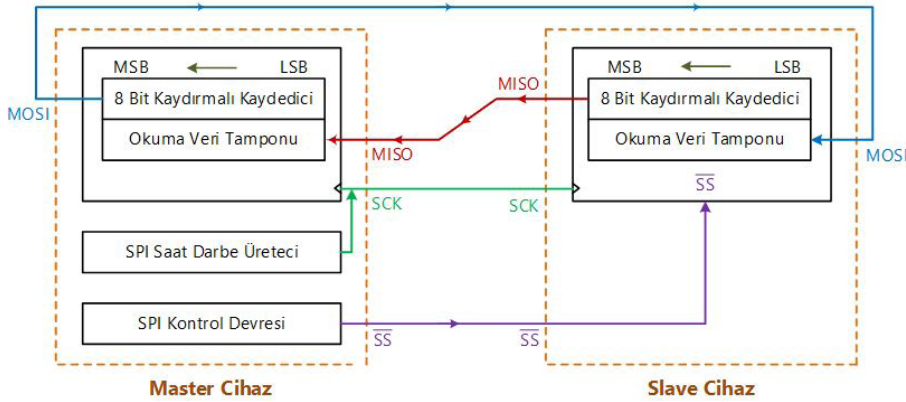
XTAL pininden uygulanan saat darbesi, frekans bölücü yardımıyla 2, 4, 8, 16, 32, 64 veya 128’e bölünerek veri aktarım hızı ayarlanabilir. Ayrıca, okuma ve yazma işlemlerinde bit kaydırma yönleri isteğe bağlı olarak ayarlanabilir.

3.5.1.1. SPI Çalışma Modları

SPI modülü, **master** veya **slave** modda çalıştırılır. SS' (Slave Select) pini, master ile slave modda çalışan cihazların birbiriyle eş zamanlı haberleşmesi için kullanılır. Cihazlar arasında iletişim kurulabilmesi için master modunda çalışan cihazın SS' pini lojik 0 düzeyine çekilerek slave modda çalışan cihaz yetkilendirilir. Cihazlar arası iletişim kurulmadığı zamanlarda master modunda çalışan cihazın SS' pininden lojik 1 gönderilir.

Master modunda çalışan cihaz, SCK pininden bir çalışma saat darbesi üretir ve slave cihaza gönderir, aynı zamanda okuma ya da yazma işlemini başlatmak üzere SS' pininden lojik 0 göndererek slave modunda çalışan cihazı yetkilendirir. Master modunda çalışan cihazdaki verinin slave modunda çalışan cihaza gönderilmesi için her iki cihazda **MOSI [Master Çıkış / Slave Giriş (Master Out/Slave In)]** pini kullanılır.

Slave modunda çalışan cihaz ise master modunda çalışan cihazın gönderdiği saat darbesi ve SS' yetkilendirme girişine göre master modunda çalışan cihaz ile eş zamanlı bir şekilde okuma ve/veya yazma işlemini gerçekleştirir. Slave modunda çalışan cihazdaki verinin master modunda çalışan cihaza gönderilmesi için her iki cihazda **MISO [Master Giriş / Slave Çıkış (Master In/Slave Out)]** pini kullanılır. SPI modülünü kullanan slave ve master modlu cihazlar arasındaki bağlantıyı gösteren şema, Görsel 3.63'te verilmiştir.



Görsel 3.63: SPI master/slave bağlantısı

Atmega328P mikrodenetleyicide yer alan SPI modülü ile ilgili pinlerini gösteren DIP soket pin şeması Görsel 3.64'te verilmiştir. Master ve slave cihazlar arasında SPI modülü üzerinden bağlantı oluşturmak için her iki cihazda karşılıklı olarak SCK, MISO, MOSI ve SS' pinleri birbirine bağlanır.

(RESET) PC6	1	28	PC5
PD0	2	27	PC4
PD1	3	26	PC3
PD2	4	25	PC2
PD3	5	24	PC1
PD4	6	23	PC0
VCC	7	22	GND
GND	8	21	AREF
(XTAL1) PB6	9	20	AVCC
(XTAL2) PB7	10	19	PB5 (SCK)
PD5	11	18	PB4 (MISO)
PD6	12	17	PB3 (MOSI)
PD7	13	16	PB2 (SS)
PB0	14	15	PB1

Görsel 3.64: Atmega328P mikrodenetleyicinin SPI modülü ile ilgili pinleri

SPI saat darbesi (SCK) işaretinin, slave modunda çalışan cihaz tarafından doğru örneklenebilmesi için SCK işaretinin düşük ve yüksek düzeyli periyotlarının en az 2 CPU saat darbesi çevrimi kadar olması beklenir.

Slave modda SS' pininin giriş olarak ayarlanması gereklidir. Master modda ise SS' pini çıkış olarak ayarlanır. MSTR bitinin 1 olması hâlinde, cihaz master modunda çalışır, 0 olması hâlinde ise slave moduna geçer.

Master/slave cihazlar arasında 8 bitlik veri aktarımı gerçekleştikten sonra, SPIF bayrağı 1 olur ve SREG kaydedicisinin I biti 1 ve SPI kesmesi etkinse program kesme rutinini çalıştırır. Kesme rutininde MSTR bitinin sıfırlanıp sıfırlanmadığı sürekli kontrol edilir. Kesme rutinindeki komutları gerçekleştirdikten sonra program kaldığı yerden çalışmaya devam eder.

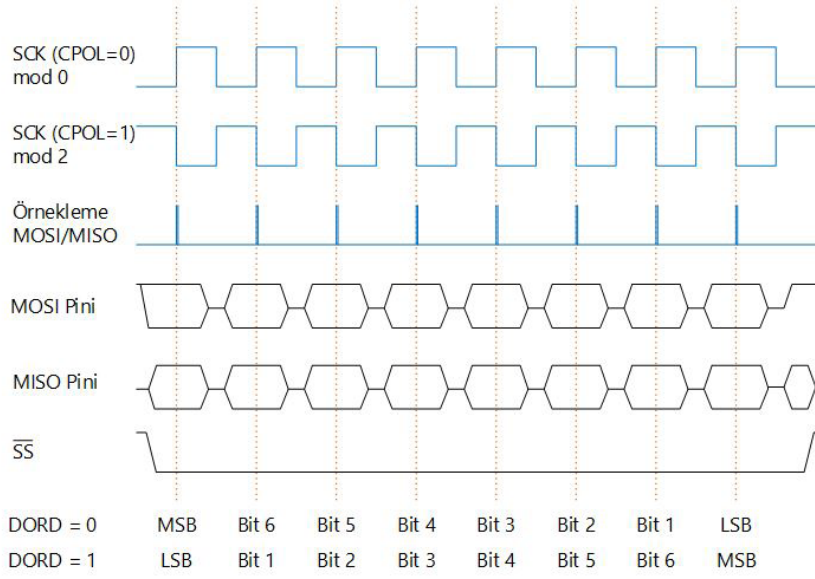
3.5.1.2. SPI Veri Modları

SCK işaretinin fazı ve polaritesi, **CPOL [saat polaritesi (Clock POLarity)]** ve **CPHA [saat fazı (Clock PHase)]** adlı kontrol bitleri ile belirlenebilmektedir. CPOL ve CPHA bitlerinin durumuna göre SCK işaretinin esas kenarı (leading edge) ve takip eden kenarının (trailing edge) değişimi Tablo 3.21'de verilmiştir.

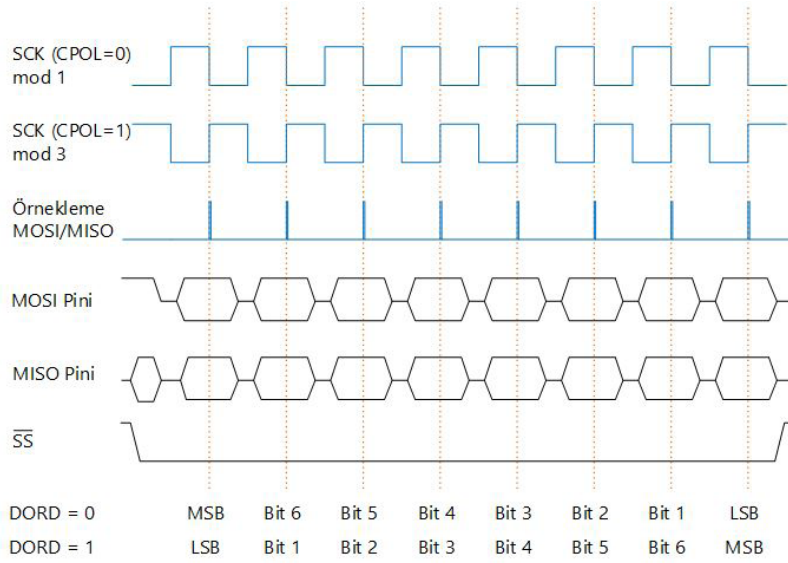
Tablo 3.21: SPI Modları

SPI Modu	CPOL	CPHA	Esas Kenar	Takip Eden Kenar
0	0	0	Örnekleme (Yükselen Kenar)	Kurulum (Düşen Kenar)
1	0	1	Kurulum (Yükselen Kenar)	Örnekleme (Düşen Kenar)
2	1	0	Örnekleme (Düşen Kenar)	Kurulum (Yükselen Kenar)
3	1	1	Kurulum (Düşen Kenar)	Örnekleme (Yükselen Kenar)

Tablo 3.21'de görüldüğü üzere CPOL kontrol biti 0 iken SCK işaretinin yükselen kenarı, 1 iken SCK işaretinin düşen kenarı ile işlem başlar. CPHA bitinin 0 olması hâlinde örnekleme SCK işaretinin yükselen kenarında, 1 olması hâlinde ise örnekleme SCK işaretinin düşen kenarında gerçekleştirilir. Bu durumun daha iyi anlaşılması için CPHA = 0 ve CPHA = 1 iken SCK işaretinin durumuna göre SPI veri aktarım formatı sırasıyla Görsel 3.65 ve 3.66'da gösterilmektedir.



Görsel 3.65: CPHA = 0 iken SPI veri aktarım formatı



Görsel 3.66: CPHA = 1 iken SPI veri aktarım formatı

3.5.1.3. SPI Kontrol Kaydedicisi

SPI kontrol kaydedicisi (SPCR), SPI işlemlerinin kontrolü için Görsel 3.67’de verilen bitlere sahiptir.

	7	6	5	4	3	2	1	0
SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

Görsel 3.67: SPCR kaydedicisi

7. Bit (SPIE): SPI kesme etkinleştirme bitidir. Bu bit ve SREG kaydedicisindeki global kesme etkinleştirme biti 1 olduğunda kesmeler etkinleştirilir. Kesme rutinine gidildiğinde, SPI kesme bayrağı SPIF sıfırlanır.

6. Bit (SPE): SPI modülünü etkinleştirmek için SPE biti 1 yapılmalıdır.

5. Bit (DORD): Veri gönderim sırasını değiştirir. DORD = 0 iken MSB bitinden LSB bitine doğru sırayla gönderim gerçekleştirilir. DORD = 1 iken LSB bitinden MSB bitine doğru sırayla gönderim gerçekleştirilir.

4. Bit (MSTR): Master/Slave seçimini gerçekleştirme bitidir. Cihaz; MSTR = 0 iken slave, MSTR = 1 iken master modda çalışır.

3. Bit (CPOL): Saat darbesi polaritesini belirler. Bu bitin etkisi Görsel 3.65 ve 3.66'dan görülebilir.

2. Bit (CPHA): Saat darbesi fazını belirler. Bu bitin etkisi Görsel 3.65 ve 3.66'dan görülebilir.

1 ve 0. Bitler (SPR1 ve SPR0): SCK saat hızını (f_{osc}) Tablo 3.22'de verilen şekilde belirlemek için kullanılır. Bu bitler, slave modunda etkin değildir. SPI2X biti, SPSR kaydedicisinde yer alır.

Tablo 3.22: SCK Osilatör Frekansı Seçenekleri

SPI2X	SPR1	SPR0	SCK Frekansı
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

3.5.1.4. SPI Veri Aktarım Durumu

SPI veri aktarım durumunun izlenmesi için SPSR (SPI durum kaydedicisi) kullanılır. SPSR, Görsel 3.68'de verilen bitlere sahiptir.

	7	6	5	4	3	2	1	0
SPSR	SPIF	WCOL	-	-	-	-	-	SPI2X

Görsel 3.68: SPSR kaydedicisi

7. Bit (SPIF): Seri aktarım işlemi bittiğinde SPIF bayrağı 1 olur. SPCR kaydedicisindeki SPIE biti ve global kesmeler etkinse kesme üretilir. SPIF biti, kesme rutinine gidilince otomatik olarak sıfırlanır veya SPIF bitine 1 yazılarak yazılım ile sıfırlanabilir.

6. Bit (WCOL): Veri aktarımı yapılırken SPDR kaydedicisine yazma yapılırsa bu bit 1 olur. WCOL bitine 1 yazılarak yazılım ile sıfırlanabilir.

0. Bit (SPI2X): Bu bit, SCK osilatör frekansı seçiminde kullanılır. Bu bitin kullanımı Tablo 3.22’de verilmiştir.



UYGULAMA

Adı:	SPI Modülünün Kullanımı	No.: 3.14
Amaç:	SPI modülünü kullanarak veri gönderme ve alma işlemlerini yapabilmek.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, SPI modülü kullanarak iki Atmega328P mikrodenetleyiciyi haberleştiren sistemi kurunuz (Mikrodenetleyicilerden biri master, diğeri slave modunda çalıştırılmalıdır. Bir mikrodenetleyiciden gönderilen veri, diğer mikrodenetleyicinin PORTD pinlerine bağlı LED’ler üzerinden izlenebilmelidir.).

Kullanılacak Araç Gereç: Tablo 3.23’te belirtilmiştir.

Tablo 3.23: 3.14 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	2 adet
Kristal	4 MHz	2 adet
Kondansatör	22 pF	4 adet
Direnç	220 Ω	8 adet
Breadboard		1 adet
Bağlantı teli	2 x 0,4 mm	1 m
LED	5 mm, Kırmızı	10 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici Programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Anahtar	Bağlantı telleri ile yapılabilir.	8 adet
Yan keski		1 adet

Uygulama Adımları:

1. Adım: Master modunda çalışacak mikrodenetleyici için Atmel Studio’da **File** menüsünden **New Project** seçeneğini seçiniz. **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **SPI_Uyg_01** yazınız..

2. Adım: Atmel Studio'da **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: Master modunda çalışan mikrodenetleyici için **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 4000000UL
#include <avr/io.h>

#define SS PORTB2 // SS' pini tanımı
#define MOSI PORTB3 // MOSI pini tanımı
#define MISO PORTB4 // MISO pini tanımı
#define SCK PORTB5 // SCK pini tanımı

void SPI_MasterInit() // SPI Master Modu Tanımlama Fonksiyonu
{
    DDRB = (1 << SS) | (1 << MOSI) | (1 << SCK); // SS', MOSI ve SCK çıkış
    SPCR = (1 << SPE) | (1 << MSTR) | (1 << SPRO); // SPI ve master etkin.
                                                    // Ön ölçekleme: fclk/16
}

void SPI_MasterTransmit(char cData) // SPI Master Modu Veri Gönderme Fonksiyonu
{
    SPDR = cData; // Gönderilen veri
    while (!(SPSR & (1 << SPIF))); // SPI bayrağı 1 olana kadar bekle.
}

int main(void)
{
    DDRD = 0x00; // Port D giriş
    PORTD = 0xFF; // Pull-up dirençleri etkin.
    SPI_MasterInit(); // SPI master modunu kur.
    while (1)
    {
        SPI_MasterTransmit(PIND); // Port D'den okunan değeri gönder.
    }
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

5. Adım: Slave modunda çalışacak mikrodenetleyici için Atmel Studio'da **File** menüsünden **New Project** seçeneğini seçiniz. **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **SPI_Uyg_02** yazınız.

6. Adım: Atmel Studio’da **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

7. Adım: Slave modunda çalışan mikrodenetleyici için **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 4000000UL
#include <avr/io.h>

#define SS PORTB2 // SS' pini tanımlama
#define MISO PORTB4 // MISO pini tanımlama

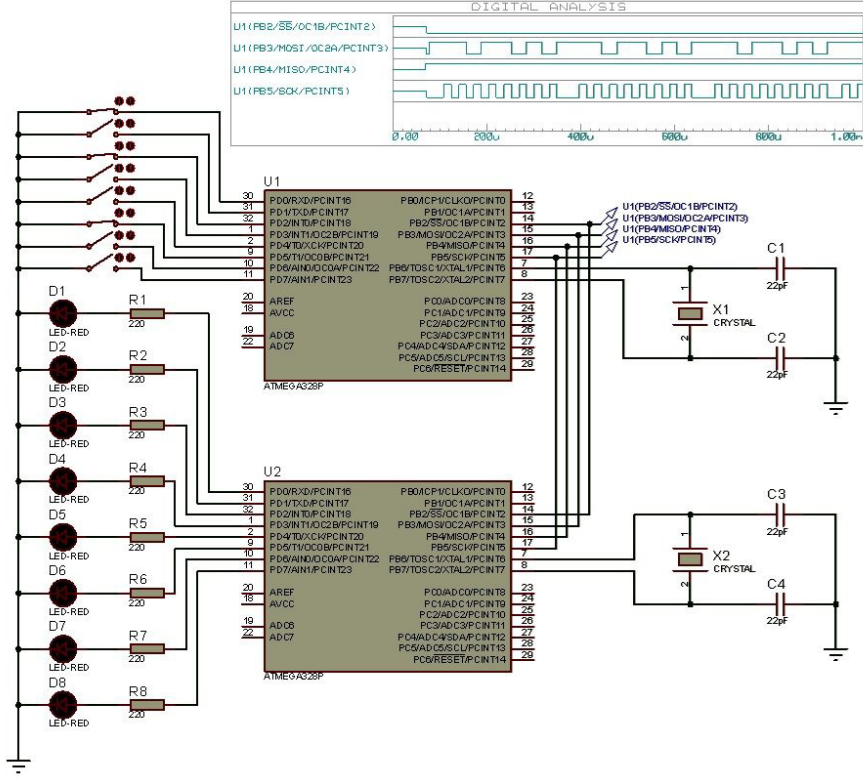
void SPI_SlaveInit() // SPI Slave Modu Tanımlama Fonksiyonu
{
    PORTB = (1 << MISO); // MISO pini çıkış
    SPCR = (1 << SPE); // SPI etkinleştir.
}

char SPI_SlaveReceive() // SPI Slave Modu Veri Alma Fonksiyonu
{
    while (!(SPSR & (1 << SPIF))); // Okuma işlemi bitene kadar bekle.
    return SPDR; // SPI veri kaydedicisi içeriğini döndür.
}

int main(void)
{
    char rData; // Alınan veri değişkeni
    DDRD = 0xFF; // PORTD tüm bitler çıkış.
    SPI_SlaveInit(); // SPI Slave modunda kur.
    while (1)
    {
        rData = SPI_SlaveReceive(); // SPI üzerinden veri al.
        PORTD = rData; // Okunan veriyi PORTD'ye yaz.
        SPSR |= (1 << SPIF); // Kesme bayrağını indir.
    }
}
```

8. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz (HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.).

9. Adım: Devre simülasyon programında Görsel 3.69’da verilen devreyi kurarak simüle ediniz. CKSEL sigortasını haricî kristal osilatör (3,0-8,0 MHz) için “1100” olarak seçiniz.



Görsel 3.69: 3.14 No.lu uygulama için simülasyon devre şeması

10. Adım: Anahtarların durumlarını değiştirerek SS, MOSI, MISO ve SCK pinlerine gerilim problemleri (*voltage probe*) bağlayıp dijital analiz (*digital analysis*) yaparak bu pinlerin çıkışlarını izleyiniz.

11. Adım: İş güvenliği tedbirlerini alarak master ve slave olarak çalışacak mikrodenetleyicileri programlayınız. Her iki mikrodenetleyici için CKSEL sigortasını haricî kristal osilatör (3,0-8,0 MHz) için “1100” olarak programlayınız.

12. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.70’te verildiği gibi kurunuz. Devreyi öğretmeninizden onay aldıktan sonra çalıştırınız.



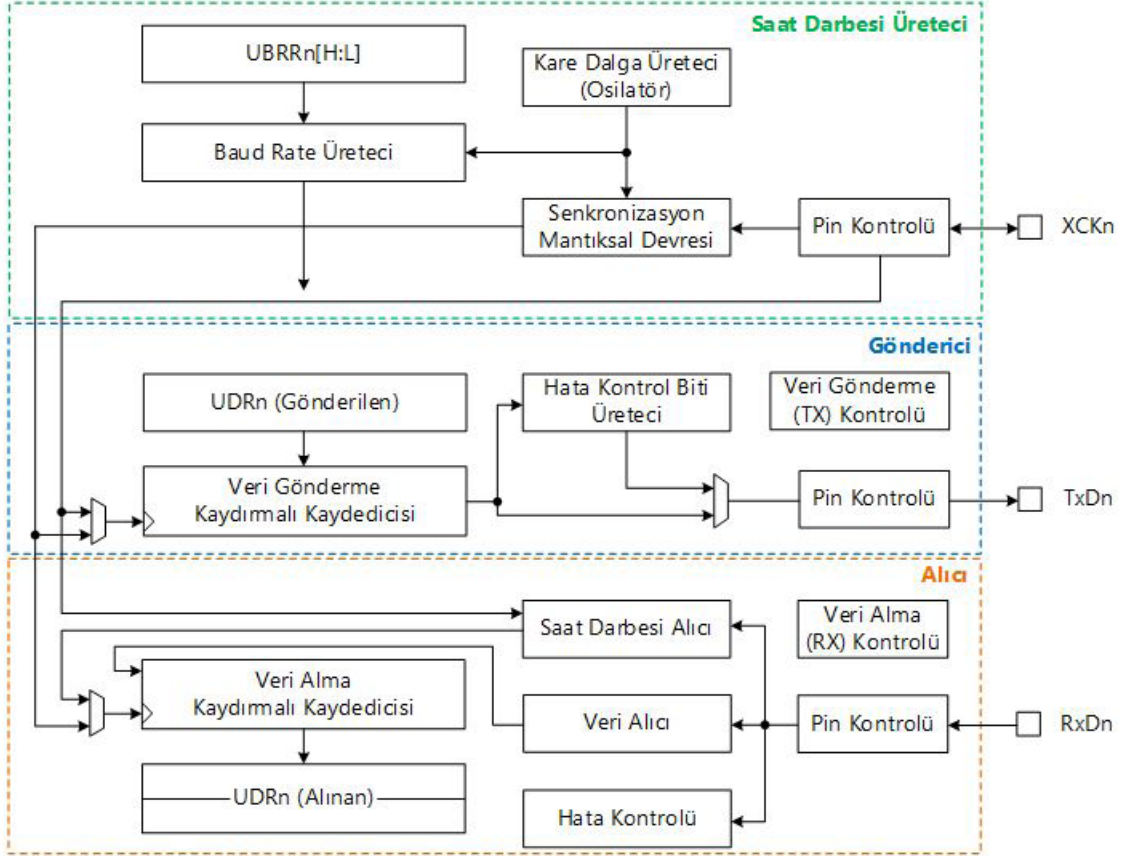
15. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Master olarak çalışan mikrodenetleyiciden gönderilen veri, slave olarak çalışan mikrodenetleyici tarafından doğru bir şekilde okunarak LED’lerde görüntülendi.		
6. Temizliğe dikkat edildi.		

3.5.2. USART Haberleşmesi

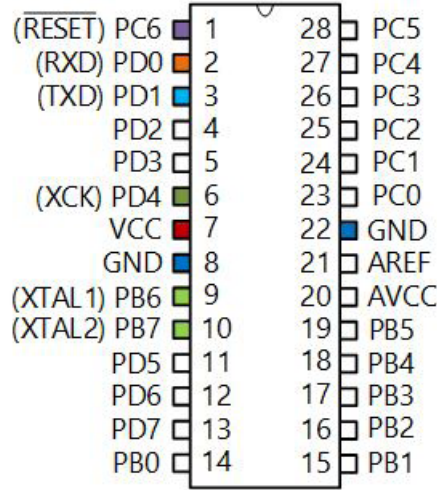
USART [Evrensel Senkron ve Asenkron Seri Alıcı ve Gönderici (Universal Synchronous and Asynchronous Serial Receiver and Transmitter)], mikrodenetleyici ile diğer mikrodenetleyiciler ve haricî aygıtlar arasında haberleşmede kullanılan diğer bir yöntemdir. USART'ın basitleştirilmiş çalışma şeması, Görsel 3.71'de verilmiştir.



Görsel 3.71: USART basitleştirilmiş çalışma şeması

Görsel 3.71'de verilen USART basitleştirilmiş çalışma şeması incelendiğinde USART modülünün saat darbesi üretici, gönderici ve alıcı olmak üzere üç alt bloktan oluştuğu görülür. Saat darbesi üretici, gönderici ve alıcı alt blokları için temel çalışma frekansını belirler. Gönderici ve alıcı alt bloklarında, verinin gönderilmesi ve alınması için kullanılan kaydırmalı kaydediciler ve gönderilen-alınan verinin saklandığı USART veri kaydedicisi [USART Data Register n (UDRn)] bulunur.

Atmega328P mikrodenetleyicide yer alan USART modülü ile ilgili pinleri gösteren DIP soket pin şeması, Görsel 3.72'de verilmiştir. Burada TXD pini üzerinden veri gönderimi, RXD pini üzerinden veri alımı yapılır. Senkron haberleşme yapılması hâlinde, master modda çalışan cihaz tarafından, slave modda çalışan cihaza XCK pini üzerinden saat darbesi işareti gönderilir. Asenkron haberleşmede XCK pininin kullanılmasına ihtiyaç duyulmaz.



Görsel 3.72: Atmega328P mikrodenetleyicinin USART modülü ile ilgili pinleri

Baud rate (baud oranı), bir haberleşme kanalı üzerinden **bir saniyede** karşı tarafa gönderilen veri oranını (işaret veya sembol değişikliklerinin sayısını) ifade eder. Baud hızı; çoğu zaman, saniyede gönderilen bit sayısı (bit/s) olarak ifade edilebilirken bazı durumlarda gönderilen bir sembol, daha fazla bilgi içerebilir. Örneğin gönderilen sembol iki bitlik bir veriye karşılık geliyorsa baud rate 4800 baud iken veri iletim hızı bu değerin iki katı kadar (9600 bit/s) olur.

Saat darbesi üretici (clock generator) alt blokunda, bir kare dalga üreticinden elde edilen saat darbesi, USART baud rate kaydedicisi [**USART Baud Rate Register n (UBRRn)**] üzerinden belirlenen hıza uygun olarak ayarlanır. Baud rate üretici çıkışında ayarlanan saat darbesi işaretinin frekansı, $f_{osc}/(UBRRn+1)$ değerine eşittir. Burada f_{osc} sistem saat darbesi işaretinin frekansıdır. Gönderici, baud rate üretici saat darbesi çıkışını, ayarlanan moda bağlı olarak /2, /8 ve /16 oranlarında ölçekleyebilir. Ayarlanan saat darbesi, veri gönderme-alma işlemlerini gerçekleştirmek üzere ilgili alt bloklara ve aynı zamanda senkron haberleşme yapılırken master modunda çalışan mikrodenetleyicinin XCKn pini üzerinden bağlı olan slave ağıta aktarılır. Slave modunda çalışan mikrodenetleyici ise gelen saat darbesini senkronize ederek (kendi ürettiği saat darbesi ile uyumlu hâle getirerek) veri gönderme-alma işlemlerini gerçekleştirmek üzere ilgili alt bloklara aktarır.

Gönderici (transmitter) alt blokunda, UDRn kaydedicisine yazılan veri, bir kaydırmalı kaydediciye aktarılır ve LSB'den MSB'ye doğru sırasıyla birer birer bitler hâlinde TXD pini üzerinden, karşı cihazın RX pinine gönderilir. Bu işlem sırasında, baud rate üreticinden elde edilen saat işareti kullanılır. Ayarlanması hâlinde gönderilen veri ile birlikte, hata kontrol bitleri de üretilerek karşı cihaza gönderilir.

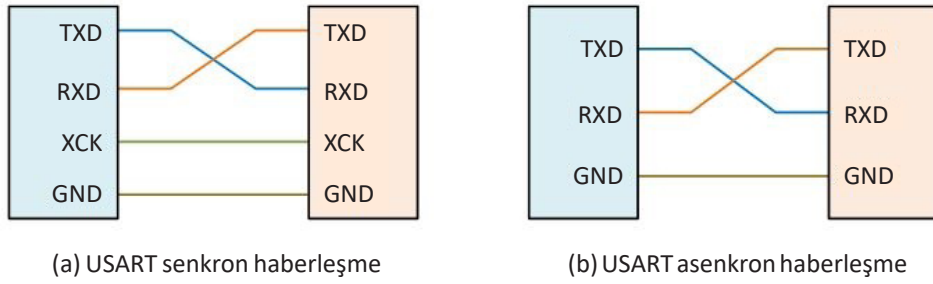
Alıcı (receiver) alt blokunda ise karşı cihazdan gönderilen ve RXD pini üzerinden elde edilen bitler, gönderim hızına uygun olarak iki cihaz arasında senkronizasyon sağlanmak suretiyle elde

edildikten sonra gerekirse hata kontrolü yapılır. Daha sonra alınan bitler, MSB'den LSB'ye doğru ilerleyen bir kaydırmalı kaydediciye sırasıyla kaydedilir. 1 baytlık veri alımı tamamlandıktan sonra alınan veri, kaydırmalı kaydediciden UDRn kaydedicisine aktarılır.

3.5.2.1. USART Çalışma Modları

Senkron (eşzamanlı) haberleşme, master modunda çalışan cihaz tarafından slave modunda çalışan cihaza, XCK pini üzerinden bir saat darbesi işaretinin gönderilmesi gereklidir.

Asenkron (eşzamanlı olmayan) haberleşme ise belirli bir saat darbesi işaretine bağlı kalınmadan istenen aralıklarla iletişim yapmayı sağlayan seçenektir ancak asenkron haberleşme, senkron haberleşmeye göre daha yavaştır. Senkron ve asenkron haberleşme, cihazların bağlantı şekli sırasıyla Görsel 3.73'te verilmiştir.



Görsel 3.73: USART cihaz bağlantıları

USART yöntemiyle hem senkron (sabit frekansta saat darbesine bağlı) hem de asenkron olarak değişik hızlarda veri aktarımı mümkündür. USART dört adet çalışma moduna sahiptir.

Normal Asenkron Modu: Normal hızda asenkron haberleşme yapılır.

Çift Kat Hızlı Asenkron Modu: Çift kat hızlı asenkron haberleşme yapılır.

Master Senkron Modu: Master modunda senkron haberleşme yapılır.

Slave Senkron Modu: Slave modunda senkron haberleşme yapılır.

USART kontrol ve durum kaydedicisi C (UCSRnC), asenkron veya senkron çalışma seçeneğini belirleyen UMSLn bitlerini içermektedir. USART kontrol ve durum kaydedicisi A (UCSRnA) ise çift hızlı (double speed) asenkron moduna geçiş yapmak için kullanılan U2Xn bitini içerir. Tablo 3.24'te her bir USART çalışma modunda baud rate (BAUD) ve UBRn değerlerinin hesaplanması için formüller verilmiştir:

Tablo 3.24: USART Çalışma Modlarına Göre BAUD ve UBRRn Değerlerinin Hesaplanması

Çalışma Modu	BAUD Değeri*	UBRRn Değeri
Normal asenkron modu (U2Xn = 0)	$\text{BAUD} = \frac{f_{\text{osc}}}{16 \times (\text{UBRRn} + 1)}$	$\text{UBRRn} = \frac{f_{\text{osc}}}{16 \times \text{BAUD}} - 1$
Çift kat hızlı asenkron modu (U2Xn = 1)	$\text{BAUD} = \frac{f_{\text{osc}}}{8 \times (\text{UBRRn} + 1)}$	$\text{UBRRn} = \frac{f_{\text{osc}}}{8 \times \text{BAUD}} - 1$
Master senkron modu	$\text{BAUD} = \frac{f_{\text{osc}}}{2 \times (\text{UBRRn} + 1)}$	$\text{UBRRn} = \frac{f_{\text{osc}}}{2 \times \text{BAUD}} - 1$

* Bu tabloda baud rate, saniyede gönderilen bit [**bits per second (bps)**] olarak tanımlanır.

Baud rate, UCSRnA kaydedicisinde yer alan U2Xn bitinin 1 yapılması ile çift kat hıza ayarlanabilir. Bu bitin ayarlanması yalnızca asenkron modda bir etkiye sahiptir. U2Xn = 1 olduğunda frekans bölme oranı 16'dan 8'e iner. Böylece USART, normale göre iki kat daha yüksek XCK frekansı ile veri gönderimi yapar. Senkron işlemlerde U2Xn biti sıfırlanmalıdır.

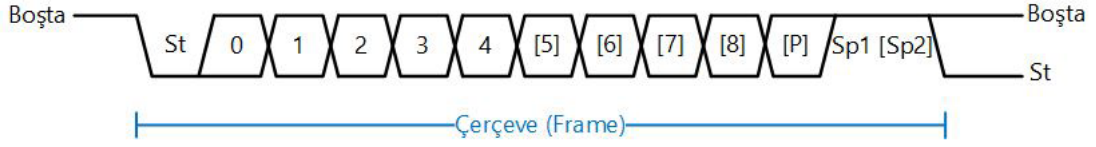
3.5.2.2. USART Çerçeve Formatları

USART modülü ile veri gönderimi sırasında kullanılan bitlerin dizilim formatına **çerçeve (frame)** adı verilir. USART protokolü olarak da adlandırılan kurallara göre aşağıda verilen bitler sırasıyla karşı tarafa iletilmelidir.

- 1 başla biti (start bit)
- 5, 6, 7, 8 veya 9 veri biti (seçime bağlı)
- 1 çift hata kontrol (even parity) veya tek hata kontrol (odd parity) biti (seçime bağlı)
- 1 veya seçime bağlı olarak 2 dur biti (stop bit)

Çerçeve, lojik 0 düzeyinde bir **başla biti** ile başlar. Başla bitini, en yüksek değerli veri biti (MSB) takip eder. Veri bitlerinin sayısı isteğe bağlı olarak 5, 6, 7, 8 veya 9 bit uzunluğunda olabilir. En düşük değerli veri biti (LSB) gönderildikten sonra, dur bitinden önce bir **çift veya tek hata kontrol biti** eklenebilir. Lojik 1 düzeyinde olan **dur bitinin** gönderilmesi ile birlikte USART, **boşta (idle)** konumuna geçer.

Görsel 3.74'te USART çerçeve formatı verilmiştir.



St: Başla biti (lojik 0)

0, 1, 2, 3, 4, 5: Veri bitleri

[6], [7] ve [8]: Seçime bağlı gönderilen veri bitleri

[P]: Seçime bağlı gönderilen hata kontrol biti

Sp1: Dur biti

[Sp2]: Seçime bağlı gönderilen ikinci dur biti

Boşta: İletişim kanalından veri transferi yok (lojik 1).

Görsel 3.74: USART çerçeve formatı

Çerçeve formatı, UCSRnB ve UCSRnC kaydedicilerinde bulunan UCSZn[2:0], UPMn[1:0] ve USBn bitleri ile ayarlanır. Burada, gönderici ve alıcının aynı format ayarında olmasına dikkat edilmelidir. Aksi hâlde veri aktarımında hatalar meydana gelir.

USART karakter uzunluğu (UCSZn[2:0]) bitleri, çerçevedeki veri biti sayısını belirler. Hata denetimi yapmak için USART hata kontrolü modu (UPMn[1:0]) bitleri etkinleştirilebilir. Seçime bağlı olarak bir veya iki dur biti gönderebilmek için USART dur biti seçim biti (USBn) kullanılabilir. Alıcı, ikinci gönderilen dur bitini dikkate almaz ancak bir çerçeve hatası olduğunda ikinci dur biti dikkate alınabilir.

3.5.2.3. Hata Kontrol Bitlerinin Hesaplanması

Hata kontrol (parity) biti, tüm bitlerin özel VEYA işlemine tabi tutulması ile bulunur. Tek hata kontrolü biti (P_{tek}) bulunurken tüm bitlerin özel VEYA işlemi sonucu 1 ile özel VEYA işlemine tabi tutulur. Çift hata kontrolü biti ($P_{çift}$) bulunurken ise tüm bitlerin özel VEYA işlemi sonucu 0 ile özel VEYA işlemine tabi tutulur. n bitten oluşan bir karakter için $i = 0, 1, \dots, n$ olmak üzere her bir bit, d_i ile ifade edildiğinde tek ve çift hata kontrol bitleri aşağıda verilen formüllerle elde edilebilir.

$$P_{tek} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1 \quad (3.1.a)$$

$$P_{çift} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0 \quad (3.1.b)$$

Özel VEYA işlemi, bitlerde değişim olması durumunda 1, bitlerin aynı kalması durumunda 0 sonucunu verir.

**ÖRNEK 16**

10111010 ile gösterilen karakterin çift ve tek hata kontrol bitlerini bulunuz.

ÇÖZÜM: (3.1.a) ve (3.1.b)'de verilen eşitlikler kullanılarak,

$$P_{\text{tek}} = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$P_{\text{çift}} = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 0 = 1 \text{ bulunur.}$$

**SIRA SİZDE**

01011100 ile gösterilen karakterin çift ve tek hata kontrol bitlerini bulunuz.

3.5.2.4. USART Giriş / Çıkış Veri Kaydedicisi n (UDRn)

USART veri gönderme ve alma tampon kaydedicileri, aynı adres bölgesini paylaşır ve **USART veri kaydedicisi (UDRn)** olarak adlandırılır. USART giriş / çıkış veri kaydedicisi, Görsel 3.75'te verilen bitlere sahiptir.

	7	6	5	4	3	2	1	0
UDRn (Okuma)	RXB[7:0]							
UDRn (Yazma)	TXB[7:0]							

Görsel 3.75: UDRn kaydedicisi

Gönderilen veri tampon kaydedicisi (TXB), UDRn kaydedicisine yazılacak verilerin önceden kaydedildiği bir yerdir. UDRn kaydedicisinin okunması sonucunda alıcı veri tampon kaydedicisinin (RXB) içeriği döndürülür.

Veri gönderme tamponuna (TXB) yalnızca UCSRnA kaydedicisinde yer alan UDREn bayrağının 1 yapılması ile veri yazılabilir. Tampona veri yazımı tamamlandıktan sonra, yazılan veri kaydırmalı kaydediciye yüklenir ve TxD pini üzerinden karşıya gönderilir.

3.5.2.5. USART Kontrol ve Durum İzleme İşlemleri

USART kontrol ve durum izleme işlemleri; UCSRnA, UCSRnB ve UCSRnC kaydedicileri ile yapılır. UCSRnA kaydedicisi genellikle bayrak bitlerini içerir. UCSRnA kaydedicisinin bitleri Görsel 3.76'da verilmiştir.

	7	6	5	4	3	2	1	0
UCSRnA	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn

Görsel 3.76: UCSRnA kaydedicisi

7. Bit (RXCn): USART veri alma işleminin tamamlandığını gösteren bayrak bitidir. Veri alma tamponunda okunacak bir veri varsa bu bayrak biti bir olur. Tampon boşken bu bit sıfırlanır.

6. Bit (TXCn): USART veri gönderme işleminin tamamlandığını gösteren bayrak bitidir. Veri gönderimi tamamlandığında TXCn biti 1 olur.

5. Bit (UDREn): USART veri kaydedicisinin boş olduğunu gösteren bayrak bitidir. UDREn = 1 olduğunda veri kaydedicisinin yeni bir veri almak için hazır olduğunu gösterir.

4. Bit (FEn): Çerçeve hatasını gösteren bayrak bitidir. FEn = 1 olduğunda çerçevenin hatalı alındığı anlaşılır.

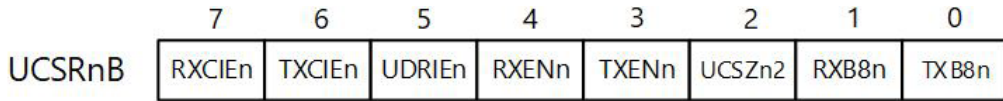
3. Bit (DORn): Veri alma tamponunun dolu olduğunu gösteren bayrak bitidir. DORn = 1 olduğunda gönderme tamponunun okunması gereklidir.

2. Bit (UPEn): USART hata kontrolünün başarısız olduğunu gösteren bayrak bitidir. UPEn = 1 olduğunda verinin doğrulamasının yapılamadığını ifade eder.

1. Bit (U2Xn): USART aktarım hızını iki katına çıkarmak için kullanılır. U2Xn = 1 olduğunda baud rate bölücü 16'dan 8'e düşer ve baud hızı iki kat artar.

0. Bit (MPCMn): Çok işlemcili haberleşme moduna geçiş sağlar. Bu bit 1 olduğunda gönderilen tüm çerçevelerin başına bir de adres verisi eklenir. Alıcı cihazlar, adresi kontrol ederek kendilerine gelen verileri kabul eder, diğer verileri dikkate almaz.

UCSRnB kaydedicisinin bitleri Görsel 3.77'de verilmiştir.



Görsel 3.77: UCSRnB kaydedicisi

7. Bit (RXCIE_n): Veri alma işlemi tamamlandığında oluşan kesmenin etkinleştirilmesini sağlayan bittir. Kesmenin etkin olması için SREG kaydedicisinde yer alan global kesme etkinleştirme (I) bitinin de 1 yapılması gereklidir.

6. Bit (TXCIE_n): Veri gönderme işlemi tamamlandığında oluşan kesmenin etkinleştirilmesini sağlayan bittir. Kesmenin etkin olması için SREG kaydedicisinde yer alan global kesme etkinleştirme (I) bitinin de 1 yapılması gereklidir.

5. Bit (UDRIE_n): USART veri kaydedicisinin boş olması hâlinde oluşan kesmenin etkinleştirilmesini sağlayan bittir. Kesmenin etkin olması için SREG kaydedicisinde yer alan global kesme etkinleştirme (I) bitinin de 1 yapılması gereklidir.

4. Bit (RXEN_n): USART veri alma işlevinin etkinleştirilmesini sağlayan bittir.

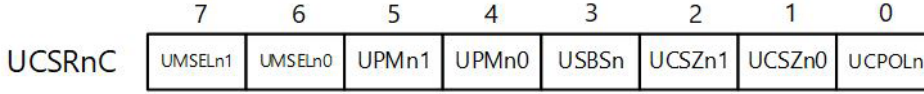
3. Bit (TXEN_n): USART veri gönderme işlevinin etkinleştirilmesini sağlayan bittir.

2. Bit (UCSZn2): Bu bit, UCSZNn[1:0] bitleri ile birlikte kullanılır. UCSZNn bitleri karakter uzunluğunu belirler.

1. Bit (RXB8n): 9 bitlik veri gönderilmesi hâlinde alınan verinin 8. bitini tutar. UDRn kaydedicisi okunmadan önce bu bit okunmalıdır.

0. Bit (TXB8n): 9 bitlik veri gönderilmesi hâlinde gönderilen verinin 8. bitini tutar. UDRn kaydedicisi okunmadan önce bu bit okunmalıdır.

UCSRnC kaydedicisinin bitleri Görsel 3.78’de verilmiştir.



Görsel 3.78: UCSRnC kaydedicisi

7 ve 6. Bitler (UMSELn[1:0]): USART’ın çalışma modunu belirler. Bit seçimleri Tablo 3.25’e göre yapılır.

Tablo 3.25: USART Çalışma Modları

UMSELn1	UMSELn0	Çalışma Modu
0	0	Asenkron USART
0	1	Senkron USART
1	0	Rezerve
1	1	Master SPI (MSPIM)

5 ve 4. Bitler (UPMn[1:0]): Hata kontrol (parity) modunu belirler. Hata kontrol bitlerinin etkinleştirilmesi ve hata kontrol biti türünü ayarlamak için kullanılır. Bit seçimleri Tablo 3.26’ya göre yapılır.

Tablo 3.26: USART Hata Modları

UMSELn1	UMSELn0	Hata Kontrol Modu
0	0	Asenkron USART
0	1	Senkron USART
1	0	Rezerve
1	1	Master SPI (MSPIM)

3. Bit (USBSn): Gönderici için dur bitinin kaç bit olacağını ayarlar. Alıcı, bu ayarlamayı dikkate almaz. Bit seçimleri Tablo 3.27’ye göre yapılır.

Tablo 3.27: Dur Biti Ayarları

USBSn	Dur Biti / Bitleri
0	1-bit
1	2-bit

2 ve 1. Bitler (UCSZn[1:0]): UCSZn[1:0] bitleri, UCSZn2 ile birlikte karakter uzunluğunu ayarlamak için kullanılır. Bit seçimleri Tablo 3.28'e göre yapılır.

Tablo 3.28: USART Hata Modları

UCSZn2	UCSZn1	UCSZn0	Karakter Uzunluğu
0	0	0	5 bit
0	0	1	6 bit
0	1	0	7 bit
0	1	1	8 bit
1	0	0	Rezerve
1	0	1	Rezerve
1	1	0	Rezerve
1	1	1	9 bit

0. Bit (UCPOLn): Saat polaritesini belirler. Yalnızca senkron modda kullanılır. XCK saat darbesinin düşen ya da yükselen kenarda okunmasını ayarlamak için kullanılır. Bit seçimleri Tablo 3.29'a göre yapılır.

Tablo 3.29: Saat Polaritesi Ayarları

UCPOLn	Gönderilen Veri Değişimi	Alınan Veri Örneklemesi
0	Yükselen XCKn Kenarı	Düşen XCKn Kenarı
1	Düşen XCKn Kenarı	Yükselen XCKn Kenarı

3.5.2.6. USART Baud Rate'in Ayarlanması

USART baud rate değerinin ayarlanması için UBRRn kaydedicisi kullanılır. 8 ile 11. bitler arasındaki veri UBRRnH, 0 ile 7. bitler arasındaki veri ise UBRRnL baytında yer alır. UBRRn kaydedicisi bitleri Görsel 3.79'da verilmiştir.

7	6	5	4	3	2	1	0
-	-	-	-	UBRRn[11:8]			
UBRRn[7:0]							

Görsel 3.79: UBRRn kaydedicisi

3.5.2.7. USART Kurulumu

USART ile veri transferi yapılabilmesi için ilk başta kurulum ayarlarının gerçekleştirilmesi gereklidir. Kurulum işlemi; baud rate, çerçeve formatı ve kullanıma göre gönderici veya alıcı etkinleştirme ayarlarının yapılmasını içermektedir. USART uygulamasında kesmeler kullanılıyorsa kurulum sırasında global kesme bayrağı sıfırlanmalıdır.

Baud rate veya çerçeve formatını değiştirmeden önce o anda veri aktarımının olmadığından emin olunmalıdır. Bunun için TXCn bayrağı kontrol edilebilir. Gönderici gönderme işlemini tamamladığında RXC bayrağı alıcı tampon belleğinde yer alan verinin okunup okunmadığını test etmek için kullanılabilir. TXCn bayrağı her veri aktarımı öncesinde sıfırlanmalıdır.

USART kurulumu için *USART_Init()* adlı fonksiyon ve bu fonksiyonun kullanımını içeren C kodu örneği aşağıda verilmiştir.

```
#define F_CPU      4000000UL           // Çalışma saat frekansı
#define BAUD      9600                // BAUD değeri
#define MYUBRR    F_CPU/16/BAUD-1    // UBRR kaydedicisi değeri

#include <avr/io.h>

void USART_Init(unsigned int ubrr)
{
    UBRR0H = (unsigned char) (ubrr >> 8); // BAUD rate ayarlama.
    UBRR0L = (unsigned char) ubrr;         // BAUD rate ayarlama.
    UCSROB = (1 << RXEN0) | (1 << TXEN0); // Gönderici ve alıcı etkinleştirme.
    UCSROC = (1 << USBS0) | (3 << UCSZ00); // Çerçeve formatı ayarlama:
                                           // 8 veri, 2 dur biti
}

int main(void)
{
    ...
    USART_Init(MYUBRR); // USART kurulumu
    ...
}
```

Çerçeve formatı parametreleri, kesmeler ve daha fazlasını ayarlama işlemleri USART kaydedicileri başlığında verilmiştir.

3.5.2.8. USART ile Veri Gönderme

USART ile veri gönderimi yapabilmek için UCSRnB kaydedicisinde yer alan gönderme etkinleştirme (Transmit Enable [**TXEN**]) bitinin 1 yapılması gereklidir. Gönderici etkinleştirildiğinde, TxD pini USART gönderme işlemi için ayarlanır. Baud rate, çalışma modu ve çerçeve formatı ayarları gönderme işleminden önce yapılmalıdır. Senkron işlem yapılacaksa XCK pininden saat darbesi çıkışı alınır.

Veri gönderme işlemi UCSROA kaydedicisinde bulunan UDRE0 bayrak bitinin durumu kontrol edilerek başlatılır. Bu bit gönderici tampon belleğinin boş olup olmama durumunu gösterir. Bu bit 1 iken UDR0 kaydedicisine gönderilecek veri yazılır. UDR0'dan kaydırmalı kaydediciye otomatik olarak aktarılan veri, gönderime hazırdır. Bu işlemden sonra gönderme işlemi başlatılır. USART veri gönderimi için kullanılan *USART_Transmit()* adlı fonksiyonu içeren C kodu örneği aşağıda verilmiştir.

```
void USART_Transmit(unsigned char data)
{
    while (!(UCSROA & (1 << UDRE0)));    // Gönderme işlemi bitene kadar bekle.
    UDR0 = data;                          // Veriyi göndermek için UDR0'a yaz.
}
```

Eğer 9 bitlik bir karakter gönderilecekse dokuzuncu bit UCSROB kaydedicisinin TXB8 bitine yazılır. 9 bitlik USART veri gönderimi için kullanılan *USART_Transmit()* adlı fonksiyonu içeren C kodu örneği aşağıda verilmiştir.

```
void USART_Transmit(unsigned char data)
{
    while (!(UCSROA & (1 << UDRE0)));    // Gönderme işlemi bitene kadar bekle.
    UCSROB &= ~(1 << TXB8);              // TXB8 bitini sıfırla.
    if (data & 0x0100)                    // 9. bit 1 ise
        UCSROB |= (1 << TXB8);          // TXB8 = 1 olarak ayarla.
    UDR0 = data;                          // Veriyi göndermek için UDR0'a yaz.
}
```

3.5.2.9. USART ile Veri Alma

USART ile veri alımı yapabilmek için USCRnB kaydedicisi içerisinde yer alan veri alma etkinleştirme (Receive Enable [**RXEN**]) bitinin 1 yapılması gereklidir. Alıcı etkinleştirildiğinde, RxD pini USART veri alma işlemi için ayarlanır. Baud rate, çalışma modu ve çerçeve formatı ayarları veri alma işleminden önce yapılmalıdır. Senkron işlem yapılacaksa XCK pinine saat darbesi girişi bağlanır.

Alıcı, veri alma işlemini başla bitinin gelmesi ile başlatır. Bu işlemden sonra her alınan bit, ayarlanan BAUD rate değerine veya XCK saat darbesi işaretine göre sırasıyla örneklenir. Dur biti alındığında, alıcı kaydırmalı kaydedicisinde bulunan veriler alıcı tampon belleğine yüklenir. Alıcı tampon belleği, UDRn kaydedicisi üzerinden okunur.

Veri alma işlemi sırasında, UCSRA kaydedicisinde yer alan RXCn bayrak bitinin durumu kontrol edilir. Bu bitin 1 olması veri alma işleminin tamamlandığını ifade eder. Veri alma işlemi tamamlandıktan sonra okunan UDRn değeri fonksiyondan geri döndürülür. Aşağıda USART veri alma işlemi için kullanılan *USART_Receive()* adlı fonksiyonu içeren C kodu örneği verilmiştir.

```
unsigned char USART_Receive()
{
    while (!(UCSRA & (1 << RXC0)));    // Veri alma işlemi bitene kadar bekle.
    return UDR0;                      // Alınan veriyi döndür.
}
```

9 bitlik bir karakter gönderildiyse 9. bit en düşük değerli bit olarak UCSROB kaydedicisinde yer alan TXB8 bitinden okunabilir.

```
unsigned char USART_Receive()
{
    while (!(UCSRA & (1 << RXC0)));           // Veri alma işlemi bitene kadar bekle.
    status = UCSRA;                             // UCSRA'yı status değişkenine ata.
    resh = UCSROB;                             // UCSRB'yi resh değişkenine ata.
    resl = UCSRC;                             // UCSRC'yi resl değişkenine ata.
    if ( status & (1 << FE0) | (1 << DOR0) | (1 << UPE0) ) // Hata varsa
        return -1;                             // Hata kodu ile çık.
    resh = (resh >> 1) & 0x01;                 // 9. biti bul, resh değişkenine at.
    return ((resh << 8) | resl);               // Alınan veriyi birleştir ve geri döndür.
}
```



UYGULAMA

Adı:	USART Modülünün Kullanımı	No.: 3.15
Amaç:	USART modülü ile haberleşme işlemlerini yapabilmek.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda, USART modülünü kullanarak iki mikrodenetleyiciyi birbiriyle haberleştiriniz.

USART asenkron modda çalıştırılmalıdır. Bir mikrodenetleyicinin PORTD pinlerinden gönderilen veri diğer mikrodenetleyicinin PORTC pinlerine bağlı LED'ler üzerinden izlenebilmelidir.

Veri gönderme ve alma işlemlerinde PORTC ve PORTD'nin ilk 6 bitini kullanınız.

Kullanılacak Araç Gereç: Tablo 3.30'da belirtilmiştir.

Tablo 3.30: 3.15 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	2 adet
Kristal	4 MHz	2 adet
Kondansatör	22 pF	4 adet
Direnç	220 Ω	12 adet
Breadboard		2 adet
Bağlantı teli	2 x 0,4 mm	1 m
LED	5 mm, Kırmızı	12 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Anahtar	Bağlantı telleri ile yapılabilir.	12 adet
Yan keski		1 adet

Uygulama Adımları:

1. Adım: İlk mikrodenetleyici için Atmel Studio'da **File** menüsünden **New Project** seçeneğini seçiniz. **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **USART_Uyg_01** yazınız.

2. Adım: Atmel Studio'da **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU      4000000UL           // Çalışma saat frekansı
#define BAUD      9600                // BAUD değeri
#define MYUBRR    F_CPU/16/BAUD-1    // UBRR kaydedicisi değeri

#include <avr/io.h>

void USART_Init (unsigned int ubrr)
{
    UBRR0H = (unsigned char) (ubrr >> 8); // BAUD rate ayarlama
    UBRR0L = (unsigned char) ubrr;        // BAUD rate ayarlama
    UCSRB = (1 << RXEN0) | (1 << TXEN0); // Gönderici ve alıcı etkinleştirme
    UCSRC = (1 << USBS0) | (3 << UCSZ00); // Çerçeve formatı ayarlama:
                                           // 8 veri, 2 dur biti
}

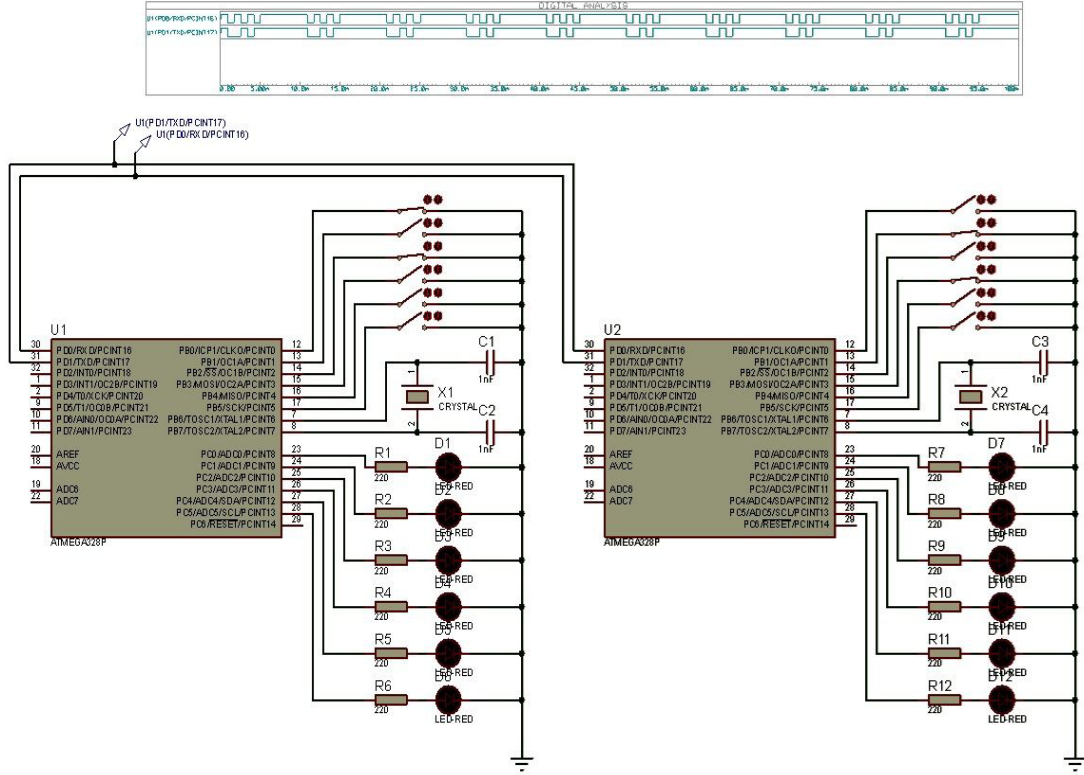
void USART_Transmit(unsigned char data) // USART veri gönderme fonksiyonu
{
    While (!(UCSR0A & (1 << UDRE0))); // Gönderme işlemi bitene kadar bekle.
    UDRO = data;                     // Veriyi göndermek için UDRO'a yaz.
}

unsigned char USART_Receive() // USART veri alma fonksiyonu
{
    while (!(UCSR0A & (1 << RXC0))); // Veri alma işlemi bitene kadar bekle.
    return UDRO;                     // Alınan veriyi döndür.
}

int main(void)
{
    char rdata;
    DDRB = 0x00; // PORTB giriş
    DDRC = 0xFF; // PORTC çıkış
    PORTB = 0xFF; // Pull-up dirençler etkin.
    USART_Init(MYUBRR); // USART kurulumu
    while (1)
    {
        USART_Transmit (PINB); // PORTB verisini USART ile gönder.
        rdata = USART_Receive(); // USART ile veri oku ve kaydet.
        PORTC = rdata;          // Okunan veriyi PORTC'de görüntüle.
    }
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

5. Adım: Devre simülasyon programında Görsel 3.80’de verilen devreyi kurarak simüle ediniz. Her iki mikrodenetleyici için aynı HEX dosyasını kullanınız. CKSEL sigortasını haricî kristal osilatör (3,0-8,0 MHz) için “1100” olarak seçiniz.

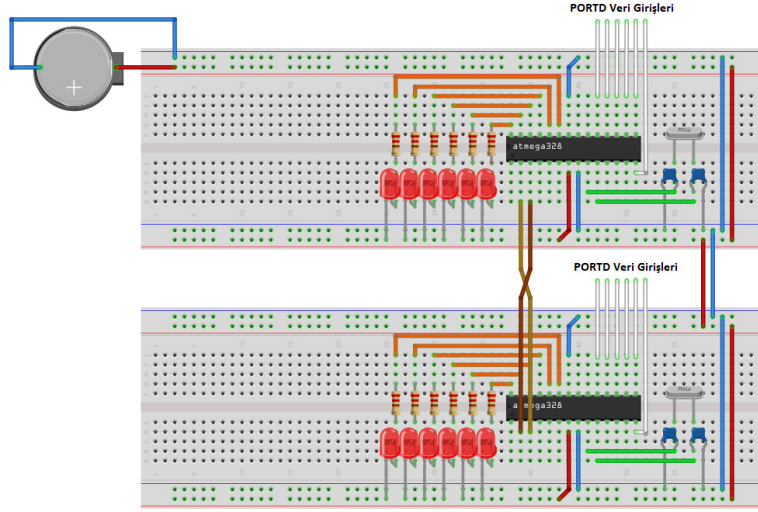


Görsel 3.80: 3.15 No.lu uygulama için simülasyon devre şeması

6. Adım: Anahtarların durumlarını değiştiriniz, RXD ve TXD pinlerine gerilim probları (*voltage probe*) bağlayıp dijital analiz (*digital analysis*) yaparak bu pinlerin çıkışlarını izleyiniz.

7. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyicileri programlayınız. Her iki mikrodenetleyici için CKSEL sigortasını haricî kristal osilatör (3,0-8,0 MHz) için “1100” olarak programlayınız.

8. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.81’de verildiği gibi kurunuz. Devreyi öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 3.81: 3.15 No.lu uygulama için breadboard üzerine kurulan devre

9. Adım: Mikrodenetleyicilerin Port D veri girişlerini +5 V veya toprağa bağlayınız. Port D girişlerinin değişimine göre LED’lerin değişimini izleyiniz. Devrenin çalışmasını öğretmeninizle birlikte değerlendiriniz.

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

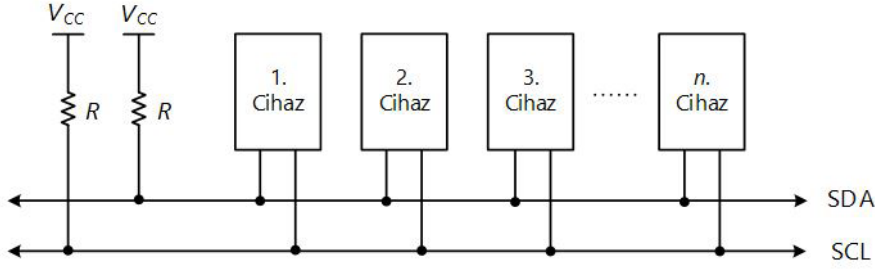
Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Bir mikrodenetleyiciden gönderilen veri, diğer mikrodenetleyici tarafından doğru bir şekilde okunarak LED’lerde görüntülendi.		
6. Temizliğe dikkat edildi.		

3.5.3. TWI Haberleşmesi

İki hatlı seri arayüz [Two-Wire Serial Interface (TWI)], haricî aygıtlar ile mikrodenetleyici arasında iletişim kurulmasını sağlayan, tümleşik devreler arası **[Inter-Integrated Circuit (I2C)]** protokolü ile uyumlu bir haberleşme yöntemidir. Kuramsal olarak 128 farklı ağıta kadar iletişim kurmaya destek veren bu yöntem, iki hat üzerinden verilerin çift yönlü ve seri olarak gönderilmesini-alınmasını sağlar. TWI haberleşmesinde kullanılan bağlantıları gösteren şema Görsel 3.82’de verilmiştir.



Görsel 3.82: TWI aygıtlar arası bağlantısı

Görsel 3.82’de görüldüğü üzere TWI haberleşmesinde iletişim, SDA ve SCL adı verilen iki hat ile sağlanmaktadır. SDA verilerin aktarımı, SCL ise saat darbesi işareti için kullanılan çift yönlü hatlardır. Bu hatlar **R** ile gösterilen pull-up dirençleri ile V_{CC} gerilimine bağlanmış, bu sayede hatlar boşken lojik 1 düzeyinde bulunmaları sağlanmıştır.

TWI haberleşmesinde iletişim, iki tür cihaz arasında sağlanır. **Master** cihaz, veri aktarımını başlatır ve sonlandırır. Aynı zamanda SCL hattı üzerinden saat darbesi işaretini üreterek gönderir. **Slave** cihaz, master cihaz tarafından adreslenen ve veri aktarımı yapılan cihazdır. Veri aktarımı, gönderme veya alma şeklinde olabilir. Veriyi gönderen cihaz **gönderici (transmitter)**, veriyi alan cihaz ise **alıcı (receiver)** olarak adlandırılır.

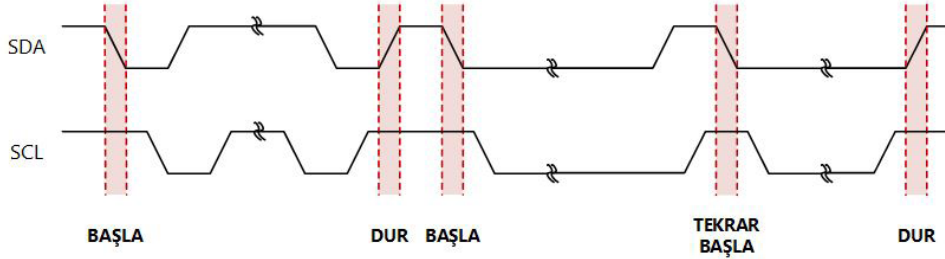
TWI haberleşmesinde verilerin aktarımı, SCL hattındaki saat darbesinin lojik 1 düzeyinde olduğu anlarda gerçekleşir. SCL hattının lojik 1’e çekilmesi sırasında SDA hattındaki veri, kararlı (stabil) olmalıdır. Aksi hâlde hatalı veri okuma gerçekleştirilir. Bu durumun tek istisnası BAŞLA ve DUR durumlarıdır.

3.5.3.1. BAŞLA ve DUR Durumları

TWI haberleşmesinde veri aktarımını başlatmak için SCL = 1 iken SDA hattı lojik 1 düzeyinden lojik 0 düzeyine düşürülür. Bu durum, **BAŞLA (START) durumu** olarak adlandırılır.

Veri aktarımını sonlandırmak için SCL = 1 iken SDA hattı, lojik 0 düzeyinden lojik 1 düzeyine çıkarılır. Bu durum ise **DUR (STOP) durumu** olarak adlandırılır.

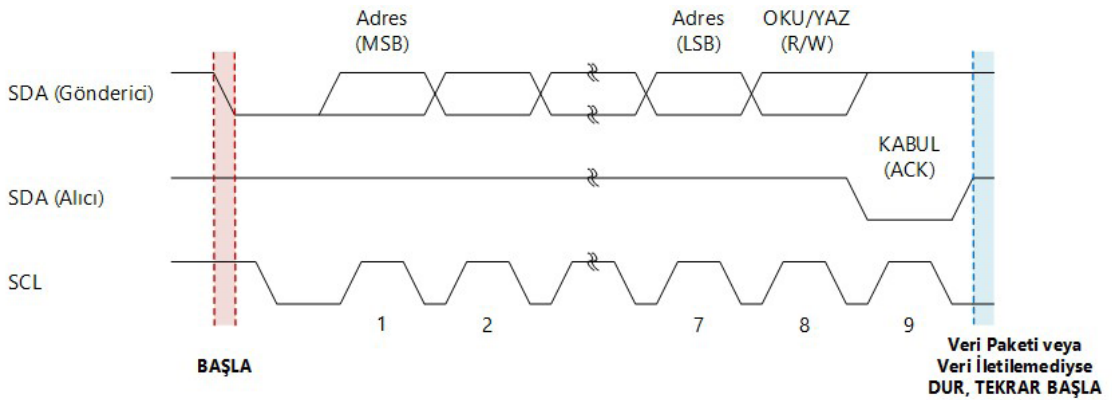
Bir iletim devam ederken BİTİR durumundan önce yeni bir aktarım başlatılmak istendiğinde yeniden BAŞLA durumu gönderilirse bu durum, **TEKRAR BAŞLA (REPEATED START)** olarak adlandırılır. BAŞLA, TEKRAR BAŞLA ve DUR durumları Görsel 3.83'te gösterilmiştir. Dikkat edilirse SDA değeri yalnızca SCL'nin lojik 1 olduğu anlarda geçerli sayılmaktadır.



Görsel 3.83: BAŞLA, TEKRAR BAŞLA ve DUR durumları

3.5.3.2. Adres Paketi Formatı

TWI haberleşmesinde hatlara bağlı her bir cihazın 7 bitlik adresi bulunur. Master cihazdan slave cihaza adresin gönderilmesi sırasında, SDA hattı üzerinden BAŞLA durumundan sonra 7 bitlik adres bilgisi gönderilir. Adres bilgisini OKU / YAZ (READ/WRITE) biti takip eder. OKU / YAZ biti, lojik 1 düzeyinde olduğunda **okuma işlemi**; lojik 0 düzeyinde olduğunda ise **yazma işlemi** gerçekleştirilir. Son olarak slave cihaz tarafından adres gönderiminin başarılı olduğunu teyit etmek amacıyla lojik 0 düzeyinde bir KABUL (ACK) biti gönderilir. Slave cihaz meşgulse ya da farklı nedenlerle master cihazın isteğine cevap veremezse SDA hattı lojik 1 düzeyinde kalır ve işlemin gerçekleştirilmediği anlaşılmış olur. Bu durumda, master cihaz tarafından DUR biti ile iletişim sonlandırılır veya TEKRAR BAŞLA biti ile yeni bir aktarım işlemi başlatılır. Adres paketi formatı, Görsel 3.84'te görülmektedir.

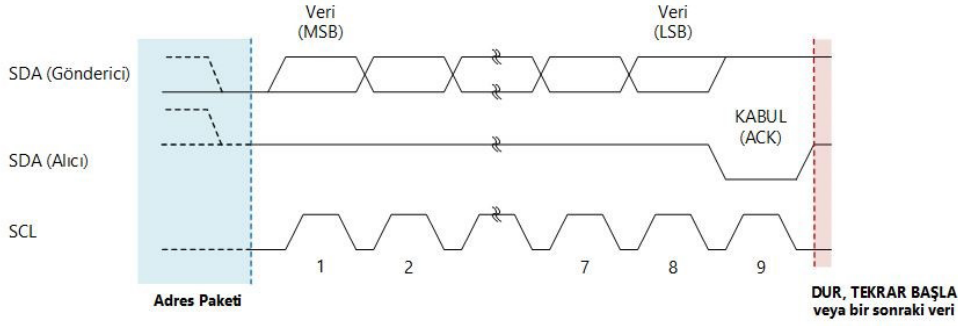


Görsel 3.84: Adres paketi formatı

Adresler, cihazlara istenen şekilde atanabilir ancak "0000 000" adresi genel çağrılar için ayrılmıştır. Bu adres ile gönderim yapıldığında hatta bağlı olan tüm cihazlara aktarım gerçekleştirilir.

3.5.3.3. Veri Paketi Formatı

TWI haberleşmesinde, veriler birer bayt olarak gönderilir. Her bir baytlık veriyi takiben alıcı cihaz tarafından bir KABUL biti gönderilir. Master cihaz; SCK hattından saat darbesi işareti, SDA hattından BAŞLA ve DUR durumlarını üretir. Alıcı, SDA hattını lojik 0 düzeyine çekerek KABUL durumu üretmezse verinin hedefe ulaşmadığı anlaşılır. Veri paketi formatı Görsel 3.85'te verilmiştir.



Görsel 3.85: Veri paketi formatı

3.5.3.4. Aktarımda Adres ve Veri Paketlerinin Birleştirilmesi

TWI haberleşmesinde bir aktarım işlemi; BAŞLA durumu, adres paketi gönderimi, bir veya daha fazla veri paketi gönderimi ve DUR durumundan oluşur. Yalnızca BAŞLA ve DUR durumlarından oluşan boş mesajlar geçersizdir. Birden fazla master cihaz, farklı SCL frekansları kullanarak aynı anda gönderim işlemine başlarsa gönderilen veriler AND işlemine tabi tutulur. Bunun sonucunda senkronizasyon tam olarak gerçekleşmediğinden veriler, doğru şekilde aktarılmaz.

Master cihazlar, göndermiş oldukları SDA işaretinin SDA hattındaki işaret ile aynı olup olmadığını sürekli denetlerler. Farklılık olması hâlinde **çekişme çözümüleme (arbitration)** durumu oluşur. Gönderdiği SDA işareti ile SDA hattındaki işaret arasında farklılık tespit eden master cihaz, master yetkilerini kaybeder ve slave olur. Diğer master cihaz veri iletimini tamamlayana kadar yeni bir aktarım işlemi başlatamaz. Bu nedenle master cihazlarda, eşzamanlı olarak birden fazla cihazı yönetmek için geliştirilmiş bir algoritma kullanılarak aktarım işlemi yönetilmelidir. TWI haberleşmesinde adres ve veri aktarım formatı Görsel 3.86'da özetlenmiştir.



Görsel 3.86: Adres ve veri aktarım formatı

Aktarımda, veri paketi sayısı isteğe bağlı olarak artırılabilir de tüm aktarım işlemlerinde aynı sayıda veri paketinin gönderilmesi olası çakışmaların ve veri kayıplarının önüne geçmek için gereklidir.

Veriyolu arayüzü birimi; BAŞLA / DUR kontrolü, gürültü bastırıcı, çekişme çözümleme (arbitrary) denetimi, adres / veri kaydırmalı kaydedicisi (TWDR) alt birimlerinden oluşmaktadır. 8 bitlik TWDR kaydedicisi, gönderilen-alınan adresleri veya verileri saklar. Ayrıca verinin teslim edilip edilmediğini gösteren 1 bitlik KABUL (ACK) bilgisinin tutulduğu ACK kaydedicisi bulunmaktadır. ACK kaydedicisine uygulama yazılımı üzerinden doğrudan erişim sağlanamaz. ACK kaydedicisi, TWI kontrol kaydedicisi (TWCR) üzerinden değiştirilebilir.

BAŞLA / DUR denetleyici; BAŞLA, DUR ve TEKRAR BAŞLA durumlarının üretilmesi ve algılanmasından sorumludur. BAŞLA / DUR denetleyici, mikrodenetleyici uyku modunda iken gelen bir adresin adres eşleştirme birimi tarafından kontrol edilmesini sağlar. Adres ilgili cihaza aitse cihaz, uyku modundan çıkarılır.

Çekişme çözümleme (arbitrary) denetimi, iki veya daha fazla master cihazın aynı anda hatları kullanıp kullanmadığını sürekli kontrol eder. Gönderilen işaret ile hatlardaki işareti karşılaştırır. İşaretlerde farklılık varsa slave moduna geçilmesini ve kontrolün bir diğer master cihaza bırakılmasını sağlar.

Gürültü bastırıcı ise alınan işarete istenmeyen bozulmaların filtrelenmesi amacıyla kullanılır. Alınan işareti sürekli kontrol ederek, bozulmalardan kaynaklı yanlış sonuçlar üretilmesini engeller.

Bit rate üretici birimi, cihaz master modunda çalışırken SCL işaretinin periyodunu kontrol eder. SCL periyodu TWI bit rate kaydedicisi (TWBR) ve TWI durum kaydedicisinde yer alan ön ölçekleme bitlerinin ayarlanmasıyla kontrol edilir. Slave modunda çalışma, bit rate ve ön ölçekleme ayarlarına bağlı değildir. Fakat CPU saat darbesi frekansının SCL frekansından (f_{scl}) en az 16 kat büyük olması gereklidir. f_{scl} aşağıdaki eşitliğe göre belirlenir.

$$f_{scl} = \frac{f_{CLK_{IO}}}{16 + 2 \cdot TWBR \cdot N}$$

Burada $f_{CLK_{IO}}$, mikrodenetleyici çalışma saat frekansını ve **N ön ölçekleme (prescaler)** değerini (1, 4, 16 veya 64 olabilir) ifade etmektedir.

Adres eşleştirme birimi, gönderilen adres değeri ile TWI adres kaydedicisi (TWAR) değerini karşılaştırır. TWAR kaydedicisi içerisinde yer alan TWI genel çağrı kabul etkinleştirme (TWGCE) biti 1 ise tüm TWAR değerine eşit olmayan adreslerin genel çağrı olup olmadığı kontrol edilir. Eşitlik durumunda, kontrol birimine bildirilir ve gerekli işlemler gerçekleştirilir.

Kontrol birimi, TWI kontrol kaydedicisi (TWCR) içerisinde yapılan ayarlara göre TWI veri yolunu kontrol eder. Aşağıda verilen olaylardan biri gerçekleştiğinde;

- TWI, BAŞLA / TEKRAR BAŞLA durumu gönderdikten sonra,
- TWI, adres paketi gönderdikten sonra,
- TWI, bir adres baytı gönderdikten sonra,
- TWI, çekişme çözümlemeyi kaybettiğinde,
- TWI, kendi slave adresini gönderdiğinde veya genel çağrıda,

- TWI, bir veri baytı aldığında,
- Slave olarak adreslenen cihazda DUR veya TEKRAR BAŞLA durumu alındığında,
- İstenmeyen BAŞLA ve DUR durumları nedeniyle hata oluştuğunda TWI kesme bayrağı (TWINT) değiştirilir.

3.5.3.6. TWI Bit Rate Değerinin Belirlenmesi

TWI bit aktarım hızının belirlenmesi için TWI bit rate kaydedicisi (**TWI Bit Rate Register [TWBR]**) kullanılır. TWBR, bit rate üreticinin bölüm faktörünü belirler. Bit rate üretici, master modunda çalışan cihazlar için SCL saat darbesi frekansını belirleyen bir frekans bölücüdür. TWBR kaydedicisinin bitleri Görsel 3.89’da verilmiştir.

	7	6	5	4	3	2	1	0
TWBR	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0

Görsel 3.89: TWBR kaydedicisi

3.5.3.7. TWI İşlemlerinin Kontrolü

TWI işlemlerinin kontrolü için TWI kontrol kaydedicisi [**TWI Control Register (TWCR)**] kullanılır. TWCR; TWI modülünü etkinleştirmek, master olarak BAŞLA durumunu oluşturmak, alıcı geri bildirimini üretmek, DUR durumunu üretmek ve veriyolu üzerine veriyi yazmak için kontrolleri sağlayan bitlere sahiptir. TWCR kaydedicisinin bitleri Görsel 3.90’da verilmiştir.

	7	6	5	4	3	2	1	0
TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE

Görsel 3.90: TWCR kaydedicisi

7. Bit (TWINT): TWI kesme bayrağı bitidir. TWI tarafından bir işlem tamamlandıktan sonra bu bit otomatik olarak 1 yapılır. SREG kaydedicisinde bulunan I biti ve TWCR kaydedicisinde bulunan TWIE biti 1 ise program, otomatik olarak TWI kesme vektörüne atlar. TWINT bayrağı 1 iken SCL pini lojik 0 düzeyine çekilir, dolayısıyla herhangi bir işlem yapılmaz. İşlemlere devam edebilmek için yazılım ile TWINT bayrağına 1 yazılarak bayrak sıfırlanmalıdır. Kescmeler etkin iken kesme rutinine gidildiğinde bu bayrak, otomatik olarak sıfırlanır. Bu bayrağın sıfırlanması işlemlerin kaldığı yerden devam etmesini sağlar, bu yüzden bayrağın sıfırlanmasından önce TWI adres kaydedicisine (TWAR), TWI durum kaydedicisine (TWSR) ve TWI veri kaydedicisine (TWDR) erişim işlemleri mutlaka tamamlanmış olmalıdır.

6. Bit (TWEA): TWI modülünde KABUL (ACK) durumlarını kontrol etmesini sağlar. TWEA = 1 ise aşağıdaki durumlarda;

1. Cihazın kendi slave adresi alınır
2. TWAR kaydedicisindeki TWGCE biti 1 iken genel bir çağrı alınır
3. Master alıcı veya slave alıcı modunda bir veri baytı alınır TWI veriyolu üzerinde KABUL işaretinin üretilmesini sağlar.

TWEA bitinin sıfırlanmasıyla cihazın sanal olarak TWI veriyolu ile bağlantısı kesilir. Adres tanımlama, TWEA = 1 yapılmasıyla tekrar etkinleştirilir.

5. Bit (TWSTA): TWI modülünde BAŞLA (START) durumlarının kontrol edilmesini sağlar. Master cihazlarda BAŞLA durumunu oluşturmak için TWSTA = 1 olmalıdır. TWI donanımı, veriyolunun uygun olup olmadığını kontrol eder ve uygunsa BAŞLA durumunu üretir. Veriyolu uygun değilse TWI modülü, veriyolunda DUR (STOP) durumunu algılayana kadar bekler ve yeni bir BAŞLA durumu oluşturur. BAŞLA durumu gönderildiğinde TWSTA yazılım ile sıfırlanmalıdır.

4. Bit (TWSTO): TWI modülünde DUR (STOP) durumlarının kontrol edilmesini sağlar. Master cihazlarda DUR durumunu oluşturmak için TWSTO = 1 olmalıdır. DUR durumu çalıştırıldığında TWSTO biti otomatik olarak sıfırlanır.

3. Bit (TWWC): Yazma sırasında çakışma durumunu gösteren bayraktır. TWINT bayrağı 0 iken TWDR kaydedicisine yazma yapılmak istendiğinde bu bayrak 1 olur. Bu bayrak, TWINT = 1 iken TWDR kaydedicisine yazma işlemi gerçekleştirildiğinde sıfırlanır.

2. Bit (TWEN): TWI modülünü etkinleştirme bitidir. TWI işlemlerini başlatmak için TWEN = 1 olmalıdır.

0. Bit (TWIE): TWI modülü kesme etkinleştirme bitidir. SREG kaydedicisinde bulunan I biti 1 iken bu bit 1 olduğunda TWI kesme isteği oluşturulur.

3.5.3.8. TWI Durumlarının İzlenmesi

TWI modülünde gerçekleşen işlemlerin durumlarının izlenmesi amacıyla TWI durum kaydedicisi (**TWI Status Register [TWSR]**) kullanılır. TWSR kaydedicisinin bitleri Görsel 3.91’de verilmiştir.

	7	6	5	4	3	2	1	0
TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0

Görsel 3.91: TWSR kaydedicisi

3-7. Bitler (TWS[7:3]): TWI modülünün durumlarını gösterir. Farklı durum kodları vardır. Okuma sırasında 5 bit durum, 2 bit ön ölçekleme değeri okunur. Okuma işleminden sonra, durum değerini elde etmek için ön ölçekleme bitleri maskelenmelidir. Durum kodları ilerleyen başlıklarda verilecektir.

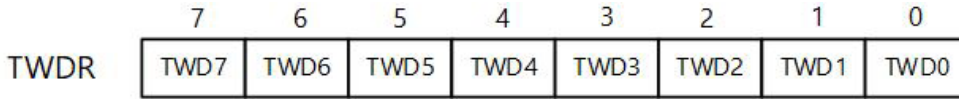
0 ve 1. Bitler (TWPS[1:0]): TWI bit rate ön ölçekleme bitleridir. Tablo 3.31'e göre ayarlanır.

Tablo 3.31: Ön Ölçekleme Değerinin Belirlenmesi

TWPS1	TWPS0	Ön Ölçekleme Değeri
0	0	1
0	1	4
1	0	16
1	1	64

3.5.3.9. TWI Modülünde Veri Gönderme ve Alma

TWI modülünde veri gönderme ve alma işlemleri için **TWI veri kaydedicisi [TWI Data Register (TWDR)]** kullanılır. TWDR, veri gönderme modunda gönderilecek veri baytını saklar. Veri alma modunda ise en son alınan veri baytını saklar. Gönderim sırasında bitler kaydırılırken TWDR'ye yazılamaz. TWINT kesme bayrağı donanım tarafından 1 yapıldığında yazma işlemi gerçekleştirilebilir. TWDR'ye ilk TWINT kesmesinden sonra erişim sağlanabilir. TWSR kaydedicisinin bitleri Görsel 3.92'de verilmiştir.

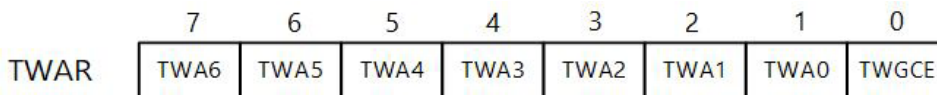


Görsel 3.92: TWDR kaydedicisi

3.5.3.10. TWI Modülünde Adresleme

TWI modülünde slave cihazlar için adres bilgileri TWI slave adres kaydedicisinde (**TWI Address Register [TWAR]**) tutulmaktadır. TWAR kaydedicisinin en büyük değerlikli 7 biti adres bilgisi için kullanılır. Master cihazlar için adres bilgisine gereksinim yoktur. Ancak birden fazla master cihaz olan sistemlerde bu adres alanı kullanılır.

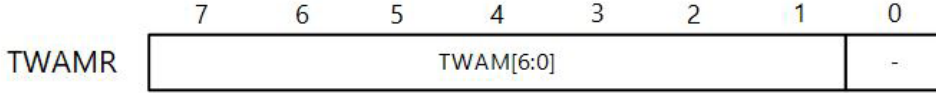
TWAR kaydedicisinin en küçük değerlikli biti (LSB) olan TWGCE biti genel çağrı oluşturmak için kullanılır. Genel çağrılarının etkinleştirildiği tüm cihazlara mesajın gönderilmesini sağlar. TWAR kaydedicisinin bitleri Görsel 3.93'te verilmiştir.



Görsel 3.93: TWAR kaydedicisi

3.5.3.11. TWI Adres Maskeleye

TWI adres maskeleye, slave cihazlarda TWAR kaydedicisinde bulunan adres bitlerinden istenenlerin maskelenmesini ve bu sayede belirli adres kriterlerine göre veri almayı sağlar. TWI adres maskeleye işlemleri, **TWI adres maskeleye kaydedici [TWI Address Mask Register (TWAMR)]** üzerinden gerçekleştirilir. Maskeleye biti 1 yapılırsa adres eşleştirme mantıksal devresi, TWAR kaydedicisinin ilgili bitlerini karşılaştırma işlemine tabi tutmaz. TWAMR kaydedicisinin bitleri Görsel 3.94'te verilmiştir.



Görsel 3.94: TWAMR kaydedicisi

3.5.3.12. Master Gönderici Modunda Haberleşme

TWI modülü ile veri aktarım işleminde, bir baytlık veri alınması veya BAŞLA durumunun gönderilmesi gibi işlem adımları sonunda TWCR kaydedicisinde yer alan TWINT kesme bayrağı 1 olur. TWINT = 1 iken program çalışmasını durdurur ve bir sonraki adım için yazılım üzerinden gerekli işlemlerin yapılmasını bekler. Gerekli işlemler yapıldıktan sonra TWINT kesme bayrağına 1 yazılarak bu bayrak biti sıfırlanır ve program kaldığı yerden devam eder. SREG kaydedicisinde yer alan global kesme etkinleştirme biti ile TWCR kaydedicisinde yer alan TWIE biti 1 yapılırsa TWINT = 1 olduğu anda program kaldığı yerden kesme vektörüne dallanır ve kesme rutini içerisine yazılan işlemler yapılır. Daha sonra program kaldığı yere geri döner ve komutları işlemeye devam eder. TWIE = 0 ise TWINT kesme bayrağının durumu aktarım işlemleri sırasında yazılım üzerinden sürekli kontrol edilmelidir. TWI modülü ile master cihazdan slave cihaza veri aktarımı sırasındaki işlem adımları aşağıda listelenmektedir.

1. TWI aktarımının ilk adımı BAŞLA durumunun gönderilmesidir. Bunun için TWCR kaydedicisine spesifik bir değer yazılır (X: Fark etmez.).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	1	0	X	1	0	X

2. BAŞLA durumu gönderildikten sonra mikrodenetleyici tarafından TWINT bayrağı 1 yapılır ve TWSR kaydedicisi, BAŞLA durumunun başarıyla gönderilip gönderilmediğini gösteren bir durum kodu ile güncellenir.

3. Uygulama yazılımı, TWSR kaydedicisine yazılan durum kodunu okur ve BAŞLA durumunun başarıyla gönderildiğini doğrular. Bunun için TWSR kaydedicisindeki değeri okur. Master gönderici modu için durum kodları ve uygulama programı tarafından yapılması gerekenler Tablo 3.32’de verilmiştir (X: Fark etmez.).

Tablo 3.32: Master Gönderici Modu İçin Durum Kodları ve Yapılması Gerekenler

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWI NT	TW EA	
0 x 08	BAŞLA durumu gönderildi.	Adres ve YAZ yüklenir.	0	0	1	X	Adres ve YAZ gönderilir, KABUL veya KABUL DEĞİL alınır.
0 x 10	TEKRAR BAŞLA durumu gönderildi.	Adres ve YAZ yüklenir.	0	0	1	X	1. Adres ve YAZ gönderilir, KABUL veya KABUL DEĞİL alınır.
		Adres ve OKU yüklenir.	0	0	1	X	2. Adres ve OKU gönderilir, master alıcı moduna geçilir.
0 x 18	Adres ve YAZ gönderildi, KABUL alındı.	Veri baytı yüklenir.	0	0	1	X	1. Veri baytı gönderilir, KABUL veya KABUL DEĞİL alınır.
		Herhangi bir işlem yapılmaz.	1	0	1	X	2. TEKRAR BAŞLA gönderilir.
		Herhangi bir işlem yapılmaz.	0	1	1	X	3. DUR gönderilir ve TWSTO bayrağı sıfırlanır.
		Herhangi bir işlem yapılmaz.	1	1	1	X	4. DUR durumunu BAŞLA durumu takip eder ve TWSTO bayrağı sıfırlanır.
0 x 20	Adres ve YAZ gönderildi, KABUL DEĞİL alındı.	Veri baytı yüklenir.	0	0	1	X	1. Veri baytı gönderilir, KABUL veya KABUL DEĞİL alınır.
		Herhangi bir işlem yapılmaz.	1	0	1	X	2. TEKRAR BAŞLA gönderilir.
		Herhangi bir işlem yapılmaz.	0	1	1	X	3. DUR gönderilir ve TWSTO bayrağı sıfırlanır.
		Herhangi bir işlem yapılmaz.	1	1	1	X	4. DUR durumunu BAŞLA durumu takip eder ve TWSTO bayrağı sıfırlanır.

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWI NT	TW EA	
0 x 28	Veri baytı gönderildi, KABUL alındı.	Veri baytı yüklenir.	0	0	1	X	1. Veri baytı gönderilir, KABUL veya KABUL DEĞİL alınır. 2. TEKRAR BAŞLA gönderilir. 3. DUR gönderilir ve TWSTO bayrağı sıfırlanır. 4. DUR durumunu BAŞLA durumu takip eder ve TWSTO bayrağı sıfırlanır.
		Herhangi bir işlem yapılmaz.	1	0	1	X	
		Herhangi bir işlem yapılmaz.	0	1	1	X	
		Herhangi bir işlem yapılmaz.	1	1	1	X	
0 x 30	Veri baytı gönderildi, KABUL DEĞİL alındı.	Veri baytı yüklenir.	0	0	1	X	1. Veri baytı gönderilir, KABUL veya KABUL DEĞİL alınır. 2. TEKRAR BAŞLA gönderilir. 3. DUR gönderilir ve TWSTO bayrağı sıfırlanır. 4. DUR durumunu BAŞLA durumu takip eder ve TWSTO bayrağı sıfırlanır.
		Herhangi bir işlem yapılmaz.	1	0	1	X	
		Herhangi bir işlem yapılmaz.	0	1	1	X	
		Herhangi bir işlem yapılmaz.	1	1	1	X	
0 x 38	Adres paketi veya veri baytlarının önderiminde çekişme durumu oluştu.	Herhangi bir işlem yapılmaz.	0	0	1	X	1. TWI serbest bırakılır ve adreslenmemiş slave moda geçilir.
		Herhangi bir işlem yapılmaz.	1	0	1	X	2. Veriyolu serbest olduğunda bir BAŞLA durumu gönderilir.

Doğrulama başarılı değilse hata kontrolü için bir işlem gerçekleştirilir. Doğrulama başarılıysa uygulama programı tarafından TWDR kaydedicisi içerisine slave cihazın adresi ve YAZ biti yazılır. TWDR hem adres hem de verilerin tutulduğu bir kaydedicidir. Gönderime devam edilmesi için TWINT biti sıfırlanmalıdır. Bunun için TWCR değeri aşağıdaki gibi ayarlanır (X: Fark etmez.).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	0	X	1	0	X

4. Adres paketi gönderildikten sonra TWINT bayrağı 1 olur ve adres paketinin başarıyla gönderildiğine dair TWSR içerisinde yer alan durum kodu güncellenir.

Önceki adımda adres paketi gönderimi başarılı olmuşsa verinin gönderilmesi için gönderilecek veri TWDR kaydedicisine yazılır. TWDR'ye yazılabilmesi için TWINT = 1 olmalıdır. TWDR kaydedicisine veri yazıldıktan sonra, veri aktarımının devam etmesi için TWINT bayrağına 1 yazılarak sıfırlanır. Veri gönderme işlemi için TWCR değeri aşağıdaki gibi ayarlanır (X: Fark etmez.).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	0	X	1	0	X

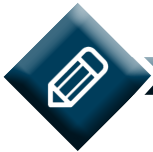
Bu işlem adımı tüm veri baytları gönderilene kadar tekrarlanır.

5. Veri paketi gönderildikten sonra TWINT bayrağı 1 olur ve veri paketinin başarıyla gönderildiğine dair TWSR içerisinde yer alan durum kodu güncellenir. Durum kodu, slave cihaz tarafından KABUL bilgisinin gönderilip gönderilmediğini de gösterir.
6. Uygulama yazılımı, veri paketinin başarıyla gönderilip gönderilmediğini tespit etmek için TWSR kaydedicisindeki durum kodu değerini kontrol eder. Son olarak DUR veya TEKRAR BAŞLA durumu gönderilir. DUR durumunun gönderilmesi için TWCR kaydedicisinin bitleri aşağıdaki gibi ayarlanır (X: Fark etmez.).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	1	X	1	0	X

İsteğe bağlı olarak TEKRAR BAŞLA durumunun gönderilmesi için TWCR kaydedicisinin bitleri aşağıdaki gibi ayarlanır (X: Fark etmez.).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	1	0	X	1	0	X



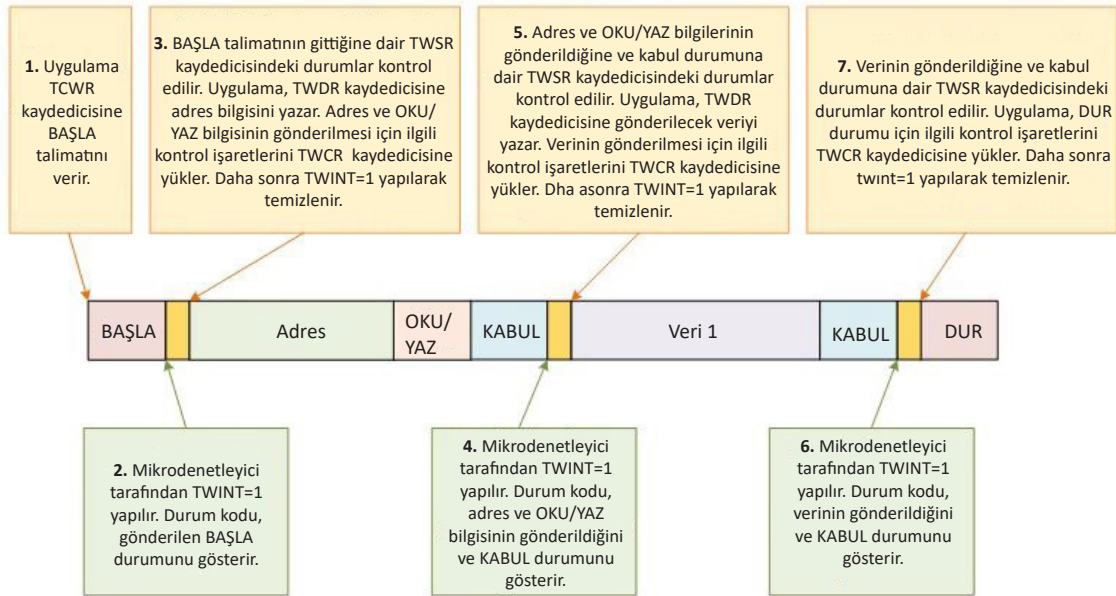
NOT

TWINT bayrağı DUR durumu gönderildikten sonra 1 olmaz. Dolayısıyla DUR durumundan sonra TWINT bayrağını sıfırlama işlemine gerek yoktur.

Sonuç olarak TWI veri transfer işlemlerinde aşağıdaki hususlara dikkat edilmelidir.

- TWI modülünde bir işlem tamamlandığında ve uygulama cevabı beklendiğinde TWINT=1 olur. TWINT = 1 iken SCL hattı sıfır yapılır, böylece gerekli hazırlıklar tamamlanmadan herhangi bir aktarım işlemi gerçekleşmez.
- TWINT = 1 olduğunda kullanıcı tarafından TWI kaydedicileri uygun değerler ile güncellenir.
- Tüm TWI kaydedicilerinin güncellenmesi ve diğer bekleyen uygulama yazılımı görevlerinin tamamlanmasının ardından, TWCR kaydedicisi uygun şekilde ayarlanarak aktarım işlevi sürdürülür.

Görsel 3.95'te TWI modülü ile master cihazdan slave cihaza veri aktarımı sırasındaki işlem adımları özetlenmiştir.



Görsel 3.95: TWI master gönderici modu veri aktarımı işlem adımları

3.5.3.13. Master Alıcı Modunda Haberleşme

Master cihaz, slave bir cihazdan veri alırken master alıcı moduna geçer. Master moduna giriş yapmak için BAŞLA durumu gönderilmelidir. Adres bilgisi ile birlikte OKU biti gönderilirse master alıcı moduna geçilir. Master alıcı modunda haberleşmek için işlem adımları aşağıda verilmiştir.

1. TWI aktarımının ilk adımı BAŞLA durumunun gönderilmesidir. Bunun için TWCR kaydedicisine spesifik bir değer yazılır (X: Fark etmez).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	1	0	X	1	0	X

2. BAŞLA durumu gönderildikten sonra mikrodenetleyici tarafından TWINT bayrağı 1 yapılır ve TWSR kaydedicisi BAŞLA durumunun başarıyla gönderilip gönderilmediğini gösteren bir durum kodu ile güncellenir.
3. Uygulama yazılımı, TWSR kaydedicisine yazılan durum kodunu okur ve BAŞLA durumunun başarıyla gönderildiğini doğrular. Bunun için TWSR kaydedicisindeki değeri okur. Gönderici için durum kodları ve uygulama programı tarafından yapılması gerekenler Tablo 3.33'te verilmiştir (X: Fark etmez).

Tablo 3.33: Master Alıcı Modu İçin Durum Kodları ve Yapılması Gerekenler

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWI NT	TW EA	
0 x 08	BAŞLA durumu gönderildi.	Adres ve OKU yüklenir.	0	0	1	X	Adres ve OKU gönderilir, KABUL veya KABUL DEĞİL alınır.
0 x 10	TEKRAR BAŞLA durumu gönderildi.	Adres ve OKU yüklenir.	0	0	1	X	1. Adres ve OKU gönderilir, KABUL veya KABUL DEĞİL alınır.
		Adres ve YAZ yüklenir.	0	0	1	X	2. Adres ve YAZ gönderilir, master alıcı moduna geçilir.
0 x 38	Adres paketi veya veri baytlarının gönderiminde çekişme durumu oluştu.	Herhangi bir işlem yapılmaz.	0	0	1	X	1. TWI serbest bırakılır ve adreslenmemiş slave moda geçilir.
		Herhangi bir işlem yapılmaz.	1	0	1	X	2. Veriyolu serbest olduğunda bir BAŞLA durumu gönderilir.
0 x 40	Adres ve OKU gönderildi, KABUL alındı.	Herhangi bir işlem yapılmaz.	0	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür.
		Herhangi bir işlem yapılmaz.	0	0	1	1	2. Veri baytı alınır ve KABUL döndürülür.
0 x 48	Adres ve OKU gönderildi, KABUL DEĞİL alındı.	Herhangi bir işlem yapılmaz.	1	0	1	X	1. TEKRAR BAŞLA gönderilir.
		Herhangi bir işlem yapılmaz.	0	1	1	X	2. DUR durumu gönderilir ve TWSTO bayrağı sıfırlanır.
		Herhangi bir işlem yapılmaz.	1	1	1	X	3. DUR durumunu takiben bir BAŞLA durumu gönderilir ve TWSTO bayrağı sıfırlanır.

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWINT	TWEA	
0 x 50	Veri baytı alındı, KABUL döndürüldü.	Veri baytı okunur.	0	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.
		Veri baytı okunur.	0	0	1	1	
0 x 58	Veri baytı alındı, KABUL DEĞİL döndürüldü.	Veri baytı okunur.	1	0	1	X	1. TEKRAR BAŞLA gönderilir. 2. DUR durumu gönderilir ve TWSTO bayrağı sıfırlanır. 3. DUR durumunu takiben bir BAŞLA durumu gönderilir ve TWSTO bayrağı sıfırlanır.
		Veri baytı okunur.	0	1	1	X	
		Veri baytı okunur.	1	1	1	X	

Eğer doğrulama başarılı değilse hata kontrolü için bir işlem gerçekleştirilir. Doğrulama başarılıysa uygulama programı tarafından TWDR kaydedicisi içerisine slave cihazın adresi ve OKU biti yazılır. TWDR hem adres hem de verilerin tutulduğu bir kaydedicidir. Adres gönderimi için TWINT biti sıfırlanmalıdır. Bunun için TWCR değeri aşağıdaki gibi ayarlanır (X: Fark etmez).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	0	X	1	0	X

- Adres paketi gönderildikten sonra TWINT bayrağı 1 olur ve adres paketinin başarıyla gönderildiğine dair TWSR içerisinde yer alan durum kodu güncellenir. Muhtemel durum kodları 0x38, 0x40 veya 0x48'dir. Bu kodların anlamları Tablo 3.33'te verilmiştir.
- Önceki adımda adres paketi gönderimi başarılı olmuşsa verinin alınması için TWINT bayrağına 1 yazılarak sıfırlanır. Veri alma işlemi için TWCR değeri aşağıdaki gibi ayarlanır (X: Fark etmez).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	0	X	1	0	X

- Okunan veri otomatik olarak TWDR kaydedicisine yazılır. TWINT bayrağı donanım tarafından 1 yapılır. TWDR kaydedici okunduktan sonra TWINT bayrağına 1 yazılarak sıfırlanır ve veri aktarımı 5 ve 6. adımlarda açıklanan şekilde devam eder.

7. Master alıcı cihaz, son bayt alındıktan sonra bir KABUL DEĞİL (NACK) göndererek karşıdaki cihaza bu durumu bildirir. Aktarım bir DUR durumu ile veya TEKRAR BAŞLA ile sonlandırılır. DUR durumunun gönderilmesi için TWCR kaydedicisinin bitleri aşağıdaki gibi ayarlanır (X: Fark etmez).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	1	X	1	0	X

İsteğe bağlı olarak TEKRAR BAŞLA durumunun gönderilmesi için TWCR kaydedicisinin bitleri aşağıdaki gibi ayarlanır (X: Fark etmez).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	1	0	X	1	0	X

TEKRAR BAŞLA durumu, master cihazın veri yolunun kontrolünü kaybetmeden slave, master gönderici veya master alıcı modlarından birine geçiş yapmasını mümkün kılar.

3.5.3.14. Slave Alıcı Modu

Master gönderici tarafından gönderilen veri baytlarını alan cihaz, slave alıcı modunda çalışır. Slave alıcı modunda işlem yapılabilmesi için aşağıdaki adımlar uygulanır:

1. Slave alıcı moduna geçiş yapmak için TWAR ve TWCR aşağıdaki gibi ayarlanmalıdır (X: Fark etmez).

TWAR değeri	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
	Slave Cihaz Adresi							X

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	0	1	0	0	X	1	0	X

TWAR kaydedicisinin en yüksek değerlikli 7 biti, slave cihaz adresini saklar. En düşük değerlikli TWGCE biti 1 yapılırsa TWI genel çağrı adresi (0x00) ile gelen aktarımlara açık olacaktır, aksi hâlde genel çağrılar dikkate alınmaz.

TWI haberleşmesinin etkinleştirilmesi için TWEN biti 1 olmalıdır. TWEA biti, cihazın kendi adresi veya genel çağrı adresi bilgisini almak için 1 yapılmalıdır. TWSTA ve TWSTO sıfır olmalıdır.

2. TWAR ve TWCR tanımlandıktan sonra, TWI modülü kendi slave adresinin (veya genel çağrı adresinin) geçtiği paketi bekler. YAZ biti gelmişse TWI modülü slave alıcı moduna, OKU biti gelmişse slave gönderici moduna geçer. TWI modülü slave alıcı modunda iken adres paketi alındıktan sonra TWINT bayrağı 1 olur ve TWSR kaydedicisinde durum kodu güncellenir. Durum kodları Tablo 3.34'te verilmiştir (X: Fark etmez).

Tablo 3.34: Slave Alıcı Modu İçin Durum Kodları

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler						Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer					
			STA	STO	TWI NT	TW EA		
0 x 60	Kendi adresi ve YAZ alındı, KABUL döndürüldü.	Herhangi bir işlem yapılmaz.	X	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.	
		Herhangi bir işlem yapılmaz.	X	0	1	1		
0 x 68	Adres paketi gönderiminde master için çekişme durumu oluştu, kendi adresi ve YAZ alındı, KABUL döndürüldü.	Herhangi bir işlem yapılmaz.	X	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.	
		Herhangi bir işlem yapılmaz.	X	0	1	1		
0 x 70	Genel çağrı adresi alındı, KABUL döndürüldü.	Herhangi bir işlem yapılmaz.	X	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.	
		Herhangi bir işlem yapılmaz.	X	0	1	1		
0 x 78	Adres paketi gönderiminde master için çekişme durumu oluştu, genel çağrı adresi alındı, KABUL döndürüldü.	Herhangi bir işlem yapılmaz.	X	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.	
		Herhangi bir işlem yapılmaz.	X	0	1	1		
0 x 80	Kendi adresi ve YAZ ile önceden adreslendi, veri alındı, KABUL döndürüldü.	Veri baytı okunur.	X	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.	
		Veri baytı okunur.	X	0	1	1		

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWI NT	TW EA	
0 x 88	Kendi adresi ve YAZ ile önceden adreslendi, veri alındı, KABUL DEĞİL döndürüldü.	Veri baytı okunur.	0	0	1	0	1. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz. 2. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır. 3. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz, veriyolu müsaitse BAŞLA durumu gönderilir. 4. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır, veriyolu müsaitse BAŞLA durumu gönderilir.
		Veri baytı okunur.	0	0	1	1	
		Veri baytı okunur.	1	0	1	0	
		Veri baytı okunur.	1	0	1	1	
0 x 90	Genel çağrı adresi ile önceden adreslendi, KABUL döndürüldü.	Veri baytı okunur.	X	0	1	0	1. Veri baytı alınır ve KABUL DEĞİL döndürülür. 2. Veri baytı alınır ve KABUL döndürülür.
		Veri baytı okunur.	X	0	1	1	
0 x 98	Genel çağrı adresi ile önceden adreslendi, KABUL DEĞİL döndürüldü.	Veri baytı okunur.	0	0	1	0	1. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz. 2. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır.
		Veri baytı okunur.	0	0	1	1	

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWI NT	TW EA	
0 x 98	Genel çağrı adresi ile önceden adreslendi, KABUL DEĞİL döndürüldü.	Veri baytı okunur.	1	0	1	0	3. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz, veriyolu müsaitse BAŞLA durumu gönderilir.
		Veri baytı okunur.	1	0	1	1	4. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır, veriyolu müsaitse BAŞLA durumu gönderilir.
0 x A0	Slave olarak adreslenmiş iken bir DUR veya TEKRAR BAŞLA durumu alındı.	Herhangi bir işlem yapılmaz.	0	0	1	0	1. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz.
			0	0	1	1	2. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır.
			1	0	1	0	3. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz, veriyolu müsaitse BAŞLA durumu gönderilir.
			1	0	1	1	4. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır, veriyolu müsaitse BAŞLA durumu gönderilir.

Durum koduna göre TWDR kaydedicisinden veri okuma işlemi gerçekleştirilir. Veri aktarımı sırasında TWEA biti 0 yapılırsa alınan veriden sonra KABUL DEĞİL (NACK) durumu gönderilir. Bu nedenle master tarafından daha sonra gönderilen veriler alınmaz.

3.5.3.15. Slave Gönderici Modu

Slave gönderici modunda, master alıcıya veri gönderimi gerçekleştirilir. Slave gönderici modunda işlem yapılabilmesi için aşağıdaki adımlar uygulanır.

1. Slave gönderici moduna geçiş yapmak için TWAR ve TWCR aşağıdaki gibi ayarlanmalıdır.

TWAR değeri	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
	Slave Cihaz Adresi							X

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	0	1	0	0	X	1	0	X

TWAR kaydedicisinin en yüksek değerlikli 7 biti, slave cihaz adresini saklar. En düşük değerlikli TWGCE biti 1 yapılırsa TWI genel çağrı adresi (0x00) ile gelen aktarımlara açık olacaktır, aksi hâlde genel çağrılar dikkate alınmaz.

TWI haberleşmesinin etkinleştirilmesi için TWEN biti 1 olmalıdır. TWEA biti, cihazın kendi adresi veya genel çağrı adresi bilgisini almak için 1 yapılmalıdır. TWSTA ve TWSTO sıfır olmalıdır.

2. TWAR ve TWCR tanımlandıktan sonra, TWI modülü kendi slave adresinin (veya genel çağrı adresinin) geçtiği paketi bekler. YAZ biti gelmişse TWI modülü slave alıcı moduna, OKU biti gelmişse slave gönderici moduna geçer. TWI modülü slave gönderici modunda iken, kendi adresinin (veya genel çağrı adresinin) bulunduğu adres paketi alındıktan sonra TWINT bayrağı 1 olur ve TWSR kaydedicisinde durum kodu güncellenir. Durum kodları Tablo 3.35'te verilmiştir (X: Fark etmez).

Tablo 3.35: Slave Gönderici Modu İçin Durum Kodları

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TW NT	TW EA	
0 x A8	Kendi adresi ve OKU alındı, KABUL döndürüldü.		X	0	1	0	1. Veri baytı gönderilir ve KABUL DEĞİL alınır.
			X	0	1	1	2. Veri baytı gönderilir ve KABUL alınır.

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler					Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer				
			STA	STO	TWI NT	TW EA	
0 x B0	Adres paketi gönderiminde master için çekişme durumu oluştu, kendi adresi ve OKU alındı, KABUL döndürüldü.	Veri baytını yükler.	X	0	1	0	1. Veri baytı gönderilir ve KABUL DEĞİL alınır.
		Veri baytını yükler.	X	0	1	1	2. Veri baytı gönderilir ve KABUL alınır.
0 x B8	TWDR kaydedicisinde yer alan veri baytı gönderildi, KABUL alındı.	Veri baytını yükler.	X	0	1	0	1. Veri baytı gönderilir ve KABUL DEĞİL alınır.
		Veri baytını yükler.	X	0	1	1	2. Veri baytı gönderilir ve KABUL alınır.
0 x C0	TWDR kaydedicisinde yer alan veri baytı gönderildi, KABUL DEĞİL alındı.	Herhangi bir işlem yapılmaz.	0	0	1	0	1. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz.
		Herhangi bir işlem yapılmaz.	0	0	1	1	2. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır.
		Herhangi bir işlem yapılmaz.	1	0	1	0	3. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz, veriyolu müsaitse BAŞLA durumu gönderilir.
		Herhangi bir işlem yapılmaz.	1	0	1	1	4. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır, veriyolu müsaitse BAŞLA durumu gönderilir.

Durum Kodu (TWSR)	Durum	Uygulama Yazılımı Tarafından Yapılması Gerekenler						Sonrasında Yapılması Gereken İşlemler
		TWDR Kaydedicisi İşlemleri	TWCR Kaydedicisine Yazılacak Değer					
			STA	STO	TWI NT	TW EA		
0 x C8	TWDR'de bulunan son veri baytı gönderildi (TWEA=0), KABUL alındı.	Herhangi bir işlem yapılmaz.	0	0	1	0	1. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz.	
		Herhangi bir işlem yapılmaz.	0	0	1	1	2. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır.	
		Herhangi bir işlem yapılmaz.	1	0	1	0	3. Adreslenmemiş slave moduna geçilir, kendi adresini ve global çağrı adresini tanımaz, veriyolu müsaitse BAŞLA durumu gönderilir.	
		Herhangi bir işlem yapılmaz.	1	0	1	1	4. Adreslenmemiş slave moduna geçilir, kendi adresini ve TWGCE = 1 ise global çağrı adresini tanır, veriyolu müsaitse BAŞLA durumu gönderilir.	

3. Önceki adımda adres paketi gönderimi başarılı olmuşsa verinin gönderilmesi için gönderilecek veri TWDR kaydedicisine yazılır. TWDR'ye yazılabilmesi için TWINT = 1 olmalıdır. TWDR kaydedicisine veri yazıldıktan sonra, veri aktarımının devam etmesi için TWINT bayrağına 1 yazılarak sıfırlanır. Veri gönderme işlemi için TWCR değeri aşağıdaki gibi ayarlanır (X: Fark etmez).

TWCR değeri	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
	1	X	0	0	X	1	0	X

Bu işlem adımı tüm veri baytları gönderilene kadar tekrarlanır. Her gönderilen veri baytından sonra TWSR kaydedicisindeki durum kodu kontrol edilir ve veri TWDR kaydedicisine yazılarak gönderimi için kesme bayrağına TWINT = 1 atanarak sıfırlanır.

Veri aktarımı sırasında TWEA biti 0 yapılırsa adres tanıma sağlanmadığından master tarafından gönderilen veriler alınmaz.



UYGULAMA

Adı:	TWI Modülünün Kullanımı	No.: 3.16
Amaç:	TWI modülü ile I2C haberleşme işlemlerini yapabilmek.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, TWI modülünü kullanarak iki mikrodenetleyiciyi birbiriyle haberleştiriniz.

Mikrodenetleyicilerden biri master diğeri slave modda, çift yönlü çalıştırılmalıdır. Bir mikrodenetleyicinin PORTB pinlerinden gönderilen veri, diğer mikrodenetleyicinin PORTD pinlerine bağlı LED'ler üzerinden izlenebilmelidir.

Veri gönderme ve alma işlemlerinde PORTB ve PORTD'nin ilk 6 bitini kullanınız.

Kullanılacak Araç Gereç: Tablo 3.36'da belirtilmiştir.

Tablo 3.36: 3.16 No.lu Uygulama İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	2 adet
Kristal	4 MHz	2 adet
Kondansatör	22 pF	4 adet
Direnç	220 Ω	12 adet
Direnç	4.7 k Ω	2 adet
Breadboard		2 adet
Bağlantı teli	2 x 0,4 mm	1 m
LED	5 mm, kırmızı	12 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Anahtar	Bağlantı telleri ile yapılabilir.	12 adet
Yan keski		1 adet

Uygulama Adımları:

1. Adım: İlk mikrodenetleyici için Atmel Studio'da **File** menüsünden **New Project** seçeneğini seçiniz. **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **TWI_I2C_Uyg_01** yazınız.

2. Adım: Atmel Studio'da **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

3. Adım: Atmel Studio'da **Solution Explorer** bölümünde **Solution** üzerine sağ tıklayarak **Add → New Item...** seçeneğini seçiniz. Karşınıza gelen **Add New Item** penceresinde **Include File** seçeneğini seçerek **Name** alanına **i2c.h** yazınız ve **Add** düğmesini tıklayınız. Açılan **i2c.h** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define START                0x08
#define RESTART              0x10
#define SLAW_TRANSMITTED_ACK 0x18
#define SLAW_TRANSMITTED_NACK 0x20
#define DATA_TRANSMITTED_ACK 0x28
#define DATA_TRANSMITTED_NACK 0x30
#define SLAR_TRANSMITTED_ARB_ACK 0x38
#define SLAR_TRANSMITTED_ACK 0x40
#define SLAR_TRANSMITTED_NACK 0x48
#define DATA_RECEIVED_ACK 0x50
#define DATA_RECEIVED_NACK 0x58
#define SLAW_RECEIVED_ACK 0x60
#define SLAW_RECEIVED_ARB_ACK 0x68
#define SLAW_RECEIVED_GC 0x70
#define SLAW_RECEIVED_ARB_GC 0x78
#define SL_DATA_RECEIVED_ACK 0x80
#define SL_DATA_RECEIVED_NACK 0x88
#define SL_DATA_RECEIVED_GC_ACK 0x90
#define SL_DATA_RECEIVED_GC_NACK 0x98
#define SLAR_RECEIVED_ACK 0xA8
#define SLAR_RECEIVED_ARB_ACK 0xB0
#define SL_DATA_TRANSMITTED_ACK 0xB8
#define SL_DATA_TRANSMITTED_NACK 0xC0
#define SL_LAST_DATA_TRANSMIT_ACK 0xC8
#define READ                1
#define WRITE                0
```

4. Adım: **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 4000000UL
#include <avr/io.h>
#include <util/delay.h>
#include "i2c.h"

int TWI_Init_MTR(void) // TWI modülünü tanıtır.
{
    TWSR = 0x00; // Durum kaydedicisini sıfırlar.
    TWBR = 0x0C; // 100 kHz baud rate (4 MHz çalışma frekansı için)
    TWCR = (1<<TWEN); // TWI etkinleştir.
    return 1; // İşlem başarılı.
}
```

```

int TWI_Send_Start_MT(void)          // BAŞLA durumu gönder.
{
    TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN) | (1 << TWEA);
    // BAŞLA durumu gönder.
    while (!(TWCR & (1 << TWINT)));
    // BAŞLA durumu gönderilene kadar bekle.
    if((TWSR & 0xF8) == START || (TWSR & 0xF8) == RESTART)
    // BAŞLA durumu gönderme işlemi başarılı mı?
        return 1;
    // BAŞLA durumu gönderme işlemi başarılı.
    else
        return 0;
    // BAŞLA durumu gönderme işlemi başarısız.
}

int TWI_Send_Address_MTR(int address, int mode)    // ADRES+OKU/YAZ gönder.
{
    TWDR = (address << 1) | mode;
    // Adres ve okuma / yazma biti.
    // Eğer mode=1 ise OKU, mode=0 ise YAZ.
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA);
    // Adresi gönder.
    while (!(TWCR & (1 << TWINT)));
    // Adres gönderilene kadar bekle.
    if(mode == 0 && ((TWSR & 0xF8) == SLAW_TRANSMITTED_ACK) ||
    ((TWSR & 0xF8) == SLAW_TRANSMITTED_NACK))
    // Yazma modunda adres gönderme işlemi başarılı mı?
        return 1;
    // Yazma modunda adres gönderme işlemi başarılı.
    if(mode == 1 && ((TWSR & 0xF8) == SLAR_TRANSMITTED_ACK) ||
    ((TWSR & 0xF8) == SLAR_TRANSMITTED_NACK) ||
    ((TWSR & 0xF8) == SLAR_TRANSMITTED_ARB_ACK))
    // Okuma modunda adres gönderme işlemi başarılı mı?
        return 2;
    // Okuma modunda adres gönderme işlemi başarılı.
    else
        return 0;
    // İşlem başarısız.
}

int TWI_Send_Data_MT(uint8_t data1)    // Veri gönder.
{
    TWDR = data1;
    // Gönderilecek veriyi TWDR'ye yaz.
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA);
    // Yazma işlemini başlat.
    while (!(TWCR & (1 << TWINT)));
    // Yazma işlemi bitene kadar bekle.
    if(((TWSR & 0xF8) == DATA_TRANSMITTED_ACK) ||
    ((TWSR & 0xF8) == DATA_TRANSMITTED_NACK))
    // Veri gönderme işlemi başarılı mı?
        return 1;
    // Veri gönderme işlemi başarılı.
    else
        return 0;
    // Değilse;
    // Veri gönderme işlemi başarısız.
}

int TWI_Read_Data_MR(uint8_t *data1, int ack) // Veri okuma fonksiyonu
{
    if(ack == 1)
    // Eğer ack=1 ise
        TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA);
    // Kontrol değerlerini ayarla (TWEA etkin.).
}

```

```

else // Değilse (0 ise)
    TWCR = (1 << TWINT) | (1 << TWEN); // Kontrol değerlerini ayarla (TWEA devre dışı.).

while (!(TWCR & (1 << TWINT))); // Okuma işlemi bitene kadar bekle.
*data1 = TWDR; // Okunan değeri döndür.
if((TWSR & 0xF8) == DATA_RECEIVED_ACK) // Veri gönderme işlemi ve KABUL alma başarılı mı?
    return 1; // Veri gönderme işlemi ve KABUL alma başarılı.
else if((TWSR & 0xF8) == DATA_RECEIVED_NACK) // Veri gönderme işlemi ve KABUL DEĞİL alma başarılı mı?
    return 2; // Veri gönderme işlemi ve KABUL DEĞİL alma başarılı.
else // Değilse
    return 0; // Veri gönderme işlemi başarısız.
}

int TWI_Send_Stop_MT(void) // DUR durumu gönderme fonksiyonu
{
    TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN); // DUR durumu gönder.
    return 1; // İşlem başarılı.
}

int main(void)
{
    DDRB = 0x00; // PORTB giriş
    PORTB = 0xFF; // Pull-up dirençleri etkin.
    DDRC = 0x00; // PORTC giriş
    PORTC = 0xFF; // Pull-up dirençleri etkin.
    DDRD = 0xFF; // PORTD çıkış
    PORTD = 0x00; // PORTD'yi sıfırla.
    uint8_t data1 = 0; // data1 değişkenini tanımla.
    TWI_Init_MTR(); // TWI Master modu kurulumu

    while (1) // Sürekli döngü
    {
        if(TWI_Send_Start_MT() == 0) // BAŞLA durumunu gönder, gönderme işlemi başarısız mı?
        {
            TWI_Send_Stop_MT(); // DUR durumunu gönder.
            continue;; // Döngü başından devam et.
        }
        if(TWI_Send_Address_MTR(0x02, READ) == 0) // Slave cihaza OKU modunda adres gönder.
            // Gönderme işlemi başarısız mı?
        {
            TWI_Send_Stop_MT(); // DUR durumunu gönder.
            continue;; // Döngü başından devam et.
        }
    }
}

```

```

_delay_us(5);    // Slave cihazda verinin yazılması için 5 us bekle.
if(TWI_Read_Data_MR(&data1, 0) == 0)
    // Slave cihazdan veri oku, okuma işlemi başarısız mı?
{
    TWI_Send_Stop_MT();    // DUR durumunu gönder.
    continue;;            // Döngü başından devam et.
}
else                    // Değilse
    PORTD = data1;
    // Okuma işlemi başarılı ise okunan veriyi PORTD'ye yaz.
if(TWI_Send_Start_MT() == 0)
    // TEKRAR BAŞLA durumu gönder.
    // Gönderme işlemi başarısız mı?
{
    TWI_Send_Stop_MT();    // DUR durumunu gönder.
    continue;;            // Döngü başından devam et.
}
if(TWI_Send_Address_MTR(0x02, WRITE) == 0)
    // Slave cihaza YAZ modunda adres gönder.
    // Gönderme işlemi başarısız mı?
{
    TWI_Send_Stop_MT();    // DUR durumunu gönder.
    continue;;            // Döngü başından devam et.
}
if(TWI_Send_Data_MT(PINB) == 0)
    // PORTB değerini slave cihaza gönder.
    // Gönderme işlemi başarısız mı?
{
    TWI_Send_Stop_MT();    // DUR durumunu gönder.
    continue;;            // Döngü başından devam et.
}
TWI_Send_Stop_MT();    // DUR durumunu gönder ve döngü başına dön.
}
}

```

5. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz.

6. Adım: Atmel Studio'da **Solution Explorer** bölümünde **Solution** üzerine sağ tıklayarak **Add → New Project...** seçeneğini seçiniz. **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **TWI_I2C_Uyg_02** yazınız.

7. Adım: Atmel Studio'da **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.

8. Adım: Atmel Studio'da **Solution Explorer** bölümünde **TWI_I2C_Uyg_02** projesi üzerine sağ tıklayarak **Add → Existing Item...** seçeneğini seçiniz. Açılan pencerede **TWI_I2C_Uyg_01** klasörüne girerek 3. adımda hazırladığınız **i2c.h** dosyasını seçtikten sonra **Add** düğmesine tıklayınız. Bu işlem sonrasında ilgili başlık dosyası, ikinci projeye de eklenmiş olacaktır.

9. Adım: Açtığınız ikinci projede **main.c** dosyası içerisine aşağıda verilen kodları yazınız.

```
#define F_CPU 4000000UL
#include <avr/io.h>
#include <util/delay.h>
#include "i2c.h"

int TWI_Init_STR(int address, int enable_GCA)           // Slave tanımlama fonksiyonu
{
    TWAR = (address << 1) | enable_GCA;                // Slave adresi tanımla.
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA);    // Slave cihaz tanımla.
                                                        // İşlem başarılı.
    return 0;
}

int TWI_Receive_Address_STR(uint8_t *address, uint8_t *mode) // Adres gönderme fonksiyonu
{
    uint8_t received;                                   // Alınan veri
    TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA);    // Adres ve mod alınabilir, KABUL gönderilir.
                                                        // Adres ve mod alma işlemi bitene kadar bekle.
    while (!(TWCR & (1 << TWINT)));
    received = TWDR;                                    // Alınan veriyi received değişkenine ata.
    *mode = received & 1;                               // Alınan veriden mod bilgisini al.
    *address = received >> 1;                           // Alınan veriden adres bilgisini al.
    if(((TWSR & 0xF8) == SLAW_RECEIVED_ACK) ||
        ((TWSR & 0xF8) == SLAW_RECEIVED_ARB_ACK)) // Adres ve yazma modu alındı mı?
        return 1;                                       // Adres ve yazma modu alma işlemi başarılı.
    else if(((TWSR & 0xF8) == SLAW_RECEIVED_GC) ||
        ((TWSR & 0xF8) == SLAW_RECEIVED_ARB_GC)) // Genel çağrı ve yazma modu alındı mı?
        return 2;                                       // Genel çağrı ve yazma modu alma işlemi başarılı.
    else if(((TWSR & 0xF8) == SLAR_RECEIVED_ACK) ||
        ((TWSR & 0xF8) == SLAR_RECEIVED_ARB_ACK)) // Adres ve okuma modu alındı mı?
        return 3;                                       // Adres ve okuma modu alma işlemi başarılı.
    else
        return 0;                                       // Değilse
                                                        // Adres ve okuma / yazma modu alma işlemi başarısız.
}

int TWI_Receive_Data_SR(uint8_t *data1, uint8_t ack) // Veri alma fonksiyonu
{
    if(ack == 0)                                         // ack = 0 ise
        TWCR = (1 << TWINT) | (1 << TWEN);            // Veri alınabilir, KABUL DEĞİL gönderilir.
    else
        TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA); // Değilse
                                                        // Veri alınabilir, KABUL gönderilir.
    while (!(TWCR & (1 << TWINT)));
                                                        // Veri alma işlemi bitene kadar bekle.
    *data1 = TWDR;                                       // Alınan veriyi kaydet.
    if(((TWSR & 0xF8) == SL_DATA_RECEIVED_ACK) ||
        ((TWSR & 0xF8) == SL_DATA_RECEIVED_NACK)) // Veri alındı mı?
        return 1;                                       // Veri alma başarılı.
```

```

else if(((TWSR & 0xF8) == SL_DATA_RECEIVED_GC_ACK) ||
((TWSR & 0xF8) == SL_DATA_RECEIVED_GC_NACK)) // Genel çağrı alındı mı?
    return 2; // Genel çağrı alma başarılı.
else // Değilse
    return 0; // Veri alma işlemi başarısız.
}
int TWI_Transmit_Data_ST(uint8_t data1) // Veri gönderme fonksiyonu
{
    TWDR = data1; // Gönderilecek veriyi TWDR'ye yaz.
    while (!(TWCR & (1<<TWINT))); // Yazma işlemi bitene kadar bekle.
    TWCR = (1<< TWINT) | (1<< TWEN); // Yazma işlemi başlat, KABUL DEĞİL alınmalı.

    if(((TWSR & 0xF8) == SL_DATA_TRANSMITTED_ACK) ||
((TWSR & 0xF8) == SL_DATA_TRANSMITTED_NACK) ||
((TWSR & 0xF8) == SL_LAST_DATA_TRANSMIT_ACK)) // Veri gönderme işlemi başarılı mı?
        return 1; // Veri gönderme işlemi başarılı.
    else // Değilse
        return 0; // Veri gönderme işlemi başarısız.
}
int main(void)
{
    DDRB = 0x00; // PORTB giriş
    PORTB = 0xFF; // Pull-up dirençleri etkin.
    DDRC = 0x00; // PORTC giriş
    PORTC = 0xFF; // Pull-up dirençleri etkin.
    DDRD = 0xFF; // PORTD çıkış
    PORTD = 0x00; // PORTD'yi sıfırla.
    uint8_t data1, address, mode; // Değişken tanımları
    TWI_Init_STR(0x02, 1); // TWI Slave modu kurulumu
    while (1) // Sürekli döngü
    {
        TWI_Init_STR(0x02, 1); // TWI Slave modu kurulumu
        switch (TWI_Receive_Address_STR(&address, &mode)) // Master'dan gönderilen adresi al.
        { // Fonksiyon dönüş değerine göre
            case 1: // 1 değeri döndüyse (Yazma modu) ve
            case 2: // 2 değeri döndüyse (Genel çağrı yazma modu)
            {
                if(TWI_Receive_Data_SR(&data1,0) == 0) // Veriyi oku, veri okuma işlemi başarısız mı?
                {
                    TWCR = (1<< TWSTO) | (1<< TWINT) | (1<< TWEN); // DUR durumu oluşturun.
                    break; // switch-case blokundan çık.
                }
            }
            else // Değilse
            {
                PORTD = data1; // Alınan veriyi PORTD'ye yaz.
                break; // switch-case blokundan çık.
            }
        }
        case 3: // 3 değeri döndüyse (Okuma modu)
        {
            if(TWI_Transmit_Data_ST(PINB) == 0) // Veri gönder, veri gönderme işlemi başarısız mı?

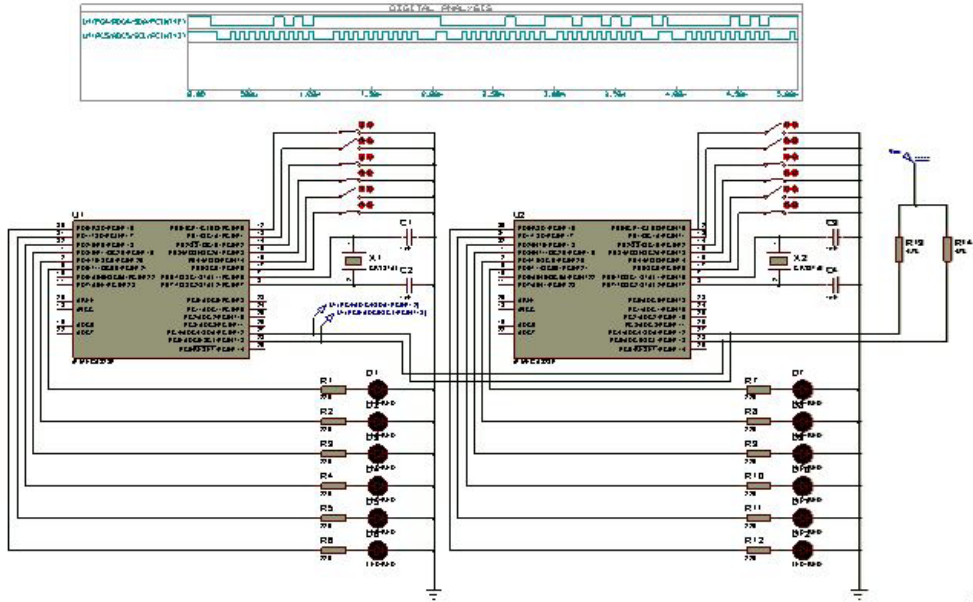
```

```

TWCR = (1 << TWSTO) | (1 << TWINT) | (1 << TWEN);
// DUR durumu oluşturun.
break; // switch-case blokundan çık.
}
case 0: // 0 değeri döndüyse (Adres alma başarısız ise)
{
TWCR = (1 << TWSTO) | (1 << TWINT) | (1 << TWEN);
// DUR durumu oluşturun.
break; // switch-case blokundan çık.
}
}
}
}

```

10. Adım: Devre simülasyon programında Görsel 3.96’da verilen devreyi kurarak simüle ediniz. Her iki mikrodenetleyici için aynı HEX dosyasını kullanınız. CKSEL sigortasını haricî kristal osilatör (3,0-8,0 MHz) için “1100” olarak seçiniz.

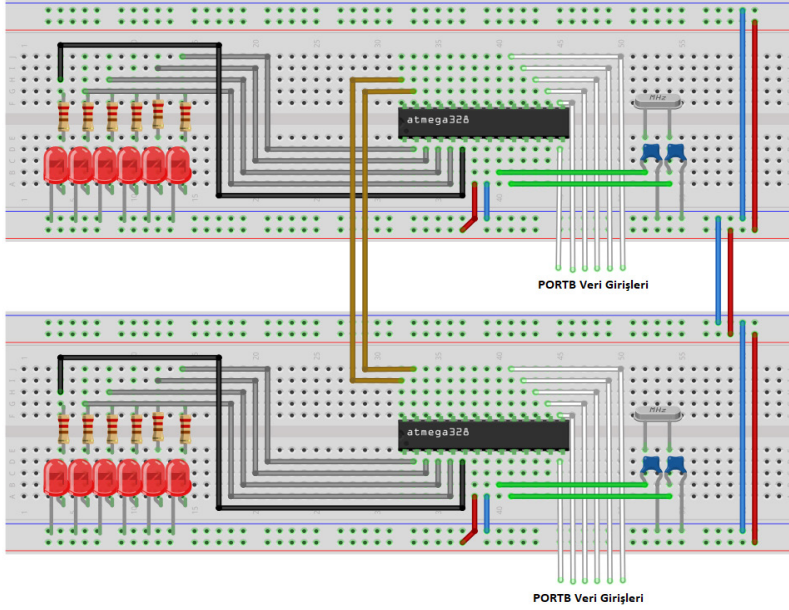


Görsel 3.96: 3.16 No.lu uygulama için simülasyon devre şeması

11. Adım: Anahtarların durumlarını değiştiriniz. SDA ve SCL pinlerine gerilim problemleri (voltage probe) bağlayıp dijital analiz (digital analysis) yaparak bu pinlerin çıkışlarını izleyiniz.

12. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyicileri programlayınız. Her iki mikrodenetleyici için CKSEL sigortasını haricî kristal osilatör (3,0-8,0 MHz) için “1100” olarak programlayınız.

13. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 3.97’de verildiği gibi kurunuz. Devreyi öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 3.97: 3.16 No.lu uygulama için breadboard üzerine kurulan devre

14. Adım: Mikrodenetleyicilerin Port B veri girişlerini +5 V veya toprağa bağlayınız. Port B girişlerinin değişimine göre LED'lerin değişimini izleyiniz. Devrenin çalışmasını öğretmeninizle birlikte değerlendiriniz.

15. Adım: Güç kaynağını kapatınız.

16. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek, başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Bir mikrodenetleyiciden gönderilen veri, diğer mikrodenetleyici tarafından doğru bir şekilde okunarak LED'lerde görüntülendi.		
6. Temizliğe dikkat edildi.		



ÖLÇME VE DEĞERLENDİRME 3

A) Aşağıda verilen cümlelerin başındaki boşluğa, cümlede yer alan ifade doğru ise “D”, yanlış ise “Y” yazınız.

1. () 7 parçalı göstergede bulunan LED’lerin anot bacakları tek noktada birleştirilmiş ise bu göstergelere, ortak katot display adı verilir.
2. () Karakter LCD’ye okuma ya da yazma işlemi gerçekleştirmek için yetki ucunun (E) önce lojik 1, sonrada lojik 0 yapılması gerekir.
3. () Çubuk (Reed) röleyi çalıştırmak için bacaklarına gerilim kaynağı bağlanır.
4. () Yük (alıcı), rölenin kontak bacaklarına bağlanır.
5. () Röleyi sürmek için anahtarlama elemanı olarak genelde transistör kullanılır.
6. () Step motorun hareketsiz olan ve sargılardan oluşan kısmına rulman adı verilir.
7. () Step motorları tam adım sürme tekniğinde, statorda bulunan bobinlerden karşılıklı olan iki bobine aynı anda enerji verilir.

B) Aşağıdaki cümlelerde bulunan boşlukları uygun kelimelerle doldurunuz.

8. LCD ilk başladığında adı verilen 15 ms’ lik kendi kendini başlatma süresine ihtiyaç duyar.
9. Manyetik röle, bobin ve olmak üzere iki ana kısımdan oluşur.
10. Yapısında yarı iletken anahtarlama elemanı bulunduran röleye, röle denir.
11. Üç fazlı elektrik motorlarını hat arızalarına karşı korumak amacıyla kullanılan röleye, röle denir.
12. İçinden akım geçen bir iletkenin manyetik alan içine yerleştirilmesiyle dönme hareketinin elde edilmesi prensibine göre çalışan makinelere denir.
13., yapısında dijital sinyalle kontrol edilebilen bobinler (stator) bulunduran ve adım adım hareket edebilen motorlardır.
14. Bir haberleşme kanalı üzerinden bir saniyede karşı tarafa gönderilen veri oranına (işaret veya sembol değişikliklerinin sayısına), adı verilir.
15. Bir USART çerçevesi (frame) en fazla bit uzunluğunda olur.
16. TWI, protokolü ile uyumlu bir haberleşme yöntemidir.

C) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz.

- 17.** Bilişim Teknolojileri alanı öğrencisi Yeliz, mikrodenetleyici ile ortak anot 7 segment display kontrolünü gerçekleştirebileceği bir devre tasarlamıştır.

Ortak anot display üzerinde E harfini gösterebilmek için displayin bağlı olduğu porta aşağıda verilen değerlerden hangisini göndermesi gerekir?

- A) 0x07 B) 0x5E C) 0x6F D) 0x77 E) 0x79

- 18.** Karakter LCD uygulaması yapmak isteyen Bilişim Teknolojileri alan öğrencisi Ahmet, karakter LCD'nin gün ışığından etkilendiğini fark etmiştir.

Projesine bir adet potansiyometre ekleyerek karakter LCD'nin kontrast ayarını değiştirmek ister. Bunun için potansiyometreyi karakter LCD'nin aşağıda verilen hangi pinine bağlaması gerekir?

- A) RS B) RW C) VBB D) VEE E) VSS

- 19.** Isıya duyarlı röle, aşağıdakilerden hangisidir?

- A) Kontaktör B) Manyetik C) Reed D) SSR E) Termik

- 20.** Elektrik motorlarının anahtarlamasında kullanılan röle, aşağıdakilerden hangisidir?

- A) Faz koruma B) Kaçak akım C) Kontaktör D) Manyetik E) Reed

- 21.** Ayarlanan süre sonunda çalışan veya duran röle, aşağıdakilerden hangisidir?

- A) Faz koruma B) Manyetik C) Reed D) Termik E) Zaman

- 22.** Haberleşme sırasında, saat darbesi (clock pulse) işaretini gönderen ve haberleşme işlemlerini yöneten cihaz, aşağıdakilerden hangisidir?

- A) Administrator B) Clock C) Master
D) Slave E) Timer

- 23.** Aşağıdakilerden hangisi SPI haberleşmesinde kullanılan pinlerden biri değildir?

- A) MISO B) MOSI C) RXD D) SCK E) SS

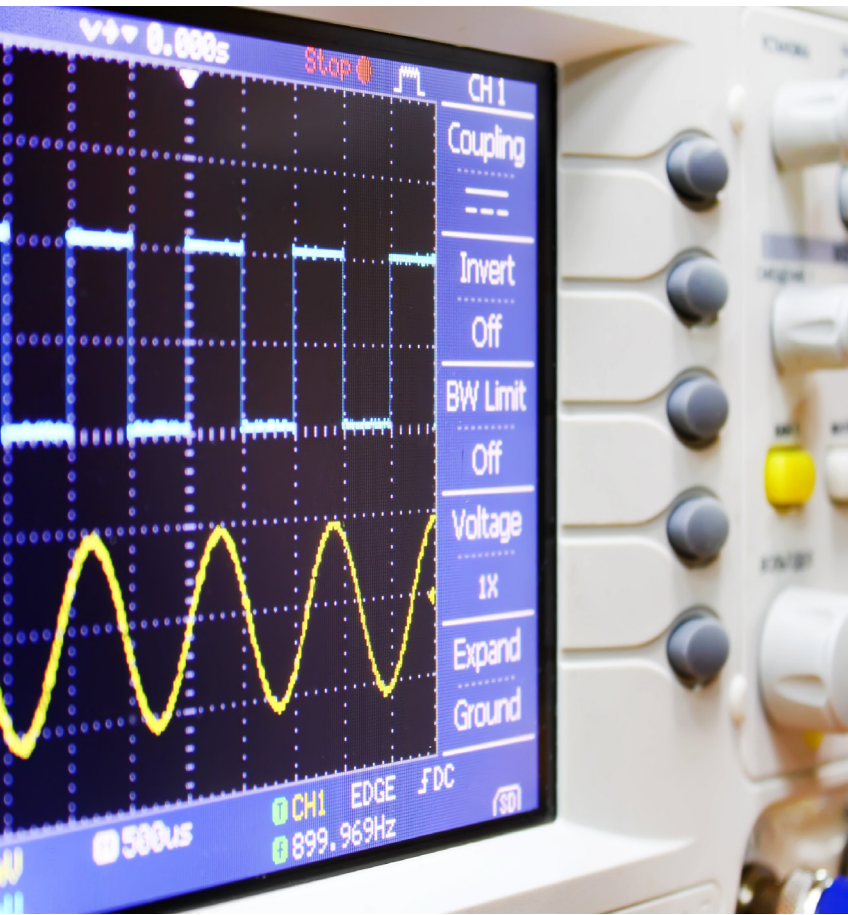
- 24.** Aşağıdakilerden hangisi SPI haberleşmesinde kullanılan pinlerden biri değildir?

- A) Çift kat hızlı asenkron modu B) Master asenkron modu
C) Master senkron modu D) Normal asenkron modu
E) Slave senkron modu

- 25.** TWI adres ve veri aktarım formatında aşağıdakilerden hangisi yer almaz?

- A) Adres B) BAŞLA durumu
C) BEKLE durumu D) DUR durumu
E) KABUL

4. ÖĞRENME BİRİMİ



**MİKRODENETLEYİCİ
İLE ANALOG
İŞLEMLER**



KONULAR

4.1. MİKRODENETLEYİCİ ADC VE DAC ÇEVİRİM KONTROLÜ

4.2. MİKRODENETLEYİCİ İLE SICAKLIK KONTROLÜ

NELER ÖĞRENECEKSİNİZ?

- ADC işlemini gerçekleştiren program
- DAC işlemini gerçekleştiren program
- Sıcaklık algılayıcıları ve türleri
- Sıcaklık algılayıcısına göre sıcaklık kontrolü

ANAHTAR KELİMELER

Analog işaret, sayısal işaret, analog / sayısal çevirici, ADC, sayısal / analog çevirici, DAC, mikrodenetleyici, algılayıcı, sıcaklık.

HAZIRLIK ÇALIŞMALARI

1. Mikrofondan alınan sesler bilgisayarın belleğine nasıl kaydedilebilir ve bu kayıt, tekrar sese nasıl dönüştürülebilir?
2. Doğru akım (DC) motorunun hızı nasıl değiştirilebilir?
3. Günlük hayatta kullandığımız hangi cihazlarda sıcaklık ölçümüne gereksinim duyulur?
4. Ortam sıcaklığını ölçmek için kullanılan analog termometreler ile dijital termometreler arasında nasıl bir fark vardır?

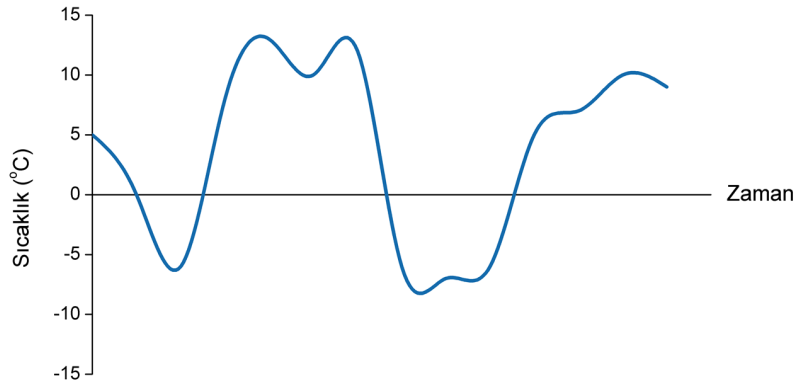
4.1. MİKRODENETLEYİCİ ADC VE DAC ÇEVİRİM KONTROLÜ

Mikrodenetleyiciler, algılayıcılardan üretilen işaretleri alarak sayısal değerlere çevirebilir. Aynı zamanda, gerilim veya akım girişli elektromekanik enerji, ışık vb. üreten haricî cihazları kontrol etmek için çıkışında işaretler üretebilir.

4.1.1. Analog ve Sayısal İşaretler

İşaretler, yaşadığımız dünyada birçok fiziksel durum veya olayın bilgisini içerir. Örneğin hava durumunu belirlemek için sıcaklık, basınç, rüzgâr hızı gibi birçok işaretten yararlanır. Telefon üzerinden görüşme yaparken akustik işaret (ses dalgaları), mikrofon yardımıyla algılanır ve elektriksel işarete dönüştürülerek karşı tarafa iletilir.

Gerçek dünyadan elde edilen sıcaklık, ses, basınç, nem gibi fiziksel büyüklüklerin ölçüm değerlerini gösteren işaretler, herhangi bir anda herhangi bir değeri alabilmekte ve zamana göre değişiklik gösterebilmektedir. Örneğin bir ortamda sürekli olarak yapılan sıcaklık ölçümleri sonucunda; zamana, konuma ve ortam koşullarına bağlı olarak birbiriyle aynı ya da birbirinden farklı ölçüm değerleri elde edilebilir. Elde edilen sıcaklık değerleri, Görsel 4.1’de görüldüğü üzere yatayda zaman eksenini, dikeyde sıcaklık eksenini şeklinde bir grafiğe dönüştürüldüğünde elde edilen işaretin zamana göre sürekli aralıktaki değerler alabilen bir karakterde olduğu görülebilir. Bu işaretlere **analog işaret** adı verilir. Gerçek dünyada fiziksel büyüklüklerin ölçümüyle elde edilen işaretler, genellikle analog işaretlerdir.



Görsel 4.1: Analog işaret

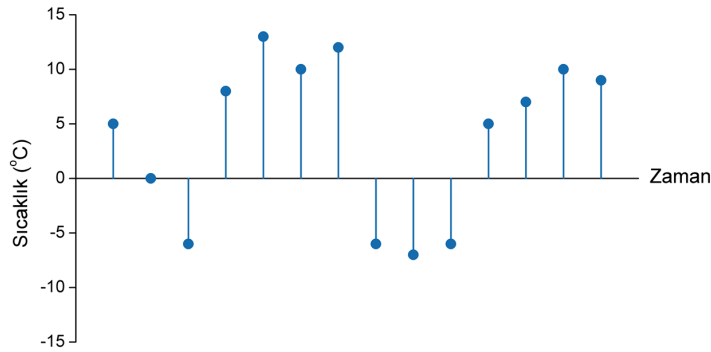
Analog işaretin genliği, işaretin üretildiği zaman diliminde, sürekli bir aralıktaki birbirinden farklı birçok değer alabilmektedir. Görsel 4.1’de görüldüğü gibi izlenen fiziksel bir büyüklüğün zamana göre değişimini gösteren fonksiyon grafiği, analog işaretlere örnektir. Görsel 4.1’de verilen işaret, sıcaklık ölçüm değerlerinin bir ortamdan sürekli olarak alınması sonucunda oluşturulduğu için **sürekli zamanlı işaret** olarak da adlandırılır. Sürekli zamanlı işaretlerin, ölçüm zaman aralığındaki tüm zaman değerlerine karşılık gelen tanımlı bir değeri bulunur.

Örneğin sıcaklığın izlendiği bir sistemde, sürekli izleme olmayıp belirli zaman aralıklarında elde edilen sıcaklık ölçüm değerleri Tablo 4.1’de verilmiştir.

Tablo 4.1: Zamana Göre Ölçülen Sıcaklık Değerleri

Zaman	Sıcaklık (°C)
1	5
2	0
3	-6
4	8
5	13
6	10
7	12
8	-6
9	-7
10	-6
11	5
12	7
13	10
14	9

Tablo 4.1’de verilen değerler, grafik üzerinde işaretlenirse Görsel 4.2’de verilen işaret oluşur. Bu işaret oluşturulurken ölçüm değerleri sürekli izlenmeyip belirli zaman aralıklarında alındığından bu işarete **ayrık zamanlı işaret** adı verilir. Ayrık zamanlı işaretler, sürekli çizgi ile gösterilmez, yalnızca ölçüm anında tanımlanan noktalar ile gösterilir.



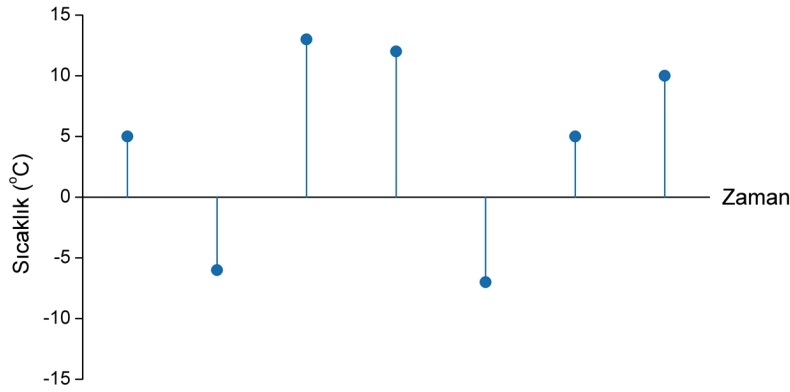
Görsel 4.2: Ayrık zamanlı işaret

Yukarıda verilen işaretler karşılaştırıldığında Görsel 4.2’de verilen ayrık zamanlı işaretin, Görsel 4.1’de verilen sürekli zamanlı işaretten, belirli zaman aralıklarında bir değer okunması (örnek alınması) ile oluştuğu görülür. Bu işleme **örnekleme** adı verilir. Örnekleme işlemi, belirlediğimiz

bir zaman diliminde bir kez yapılır. Mesela sıcaklığın algılandığı bir sistemde, sıcaklık ölçüm aralığı saniyede bir kez olarak seçilebilir. İki örnekleme arasında geçen bu süre, örnekleme periyodu (T_s) olarak adlandırılır. Örnekleme periyodunun standart birimi saniyedir.

Görsel 4.2’de verilen ayrık zamanlı işaretin noktaları, sırasıyla çizgi ile birleştirildiğinde Görsel 4.1’deki sürekli zamanlı işarete benzer bir işaret elde edilir. Ölçüm değerlerinden alınan örnek sıklığı arttıkça sürekli zamanlı işareti daha iyi temsil eden bir işaret elde edilecektir.

Örneğin Tablo 4.1’de verilen ölçümlerden her iki değerden yalnızca biri alınarak çizilecek grafik, Görsel 4.3’teki gibi oluşur. Örnek sayısı azaltılarak örnekleme periyodu artırılmış olur yani yukarıdaki örneğe göre sıcaklık, belirli bir zaman aralığında iki kez yerine bir kez okunmuş olur. Bu durumda alınan örnek sayısı daha az olduğundan Görsel 4.3’te verilen ayrık zamanlı işaretin Görsel 4.1’de verilen sürekli zamanlı işareti temsil yeteneği giderek azalır.



Görsel 4.3: Örneklem periyodu Görsel 4.2’deki işareten iki kat büyük olan ayrık zamanlı işaret



SIRA SİZDE

Görsel 4.2 ve 4.3’te verilen ayrık zamanlı işaretlerin değer noktalarını sırasıyla bir çizgi ile birleştiriniz. Birleştirme sonucunda elde ettiğiniz grafikleri Görsel 4.1’de verilen sürekli zamanlı işaret ile karşılaştırınız. Karşılaştırma sonucunda Görsel 4.2 ve 4.3’te verilen hangi ayrık zamanlı işaret, Görsel 4.1’de verilen sürekli zamanlı işareti daha iyi temsil etmektedir? Nedenini belirtiniz.

Örnekleme işleminin sıklığını göstermek için genellikle örnekleme frekansı kavramı kullanılır. **Örnekleme frekansı (f_s)**, birim zamanda alınan örnek sayısını ifade eder. Örnekleme frekansı aşağıdaki formül ile basitçe ifade edilebilir:

$$f_s = 1 / T_s$$

Burada T_s örnekleme periyodu olup saniye (s) cinsindendir. Böylece örnekleme frekansı, örnekleme periyodunun tersi alınarak bulunabilir. Örnekleme frekansının birimi **hertzdir (Hz)**.

**NOT**

Nyquist Kriteri veya Shannon Kuramına göre “Bir analog işaretin örnekleme işleminde veri kaybının yaşanmaması için örnekleme frekansı (f_s), analog işaretin frekansının (f_a) iki katından büyük olmalıdır ($f_s > 2f_a$). Aksi hâlde örnekleme işlemi sonucunda elde edilen işaret, analog işareti tam anlamıyla temsil etmez.”

**ÖRNEK 1**

Bir serada özel bir bitkinin yetiştirildiği toprağın nem değerleri, bir nem algılayıcısı yardımıyla her 100 ms’de bir kez okunmaktadır. Örnekleme frekansını hesaplayınız.

Verilenler: Örnekleme periyodu $T_s = 100 \text{ ms} = 100 \cdot 10^{-3} \text{ s}$

İstenen : Örnekleme frekansı (f_s)

ÇÖZÜM: Örnekleme işlemi her 100 ms’de bir kez yapıldığında

$$f_s = 1 / T_s = 1 / (100 \cdot 10^{-3}) = 10 \text{ Hz bulunur.}$$

Bu sonuç, örnekleme işleminde **her saniyede 10 örnek** alındığını ifade eder.

**SIRA SİZDE**

Bir DC motordan her saniyede 1 000 akım ölçüm değeri alınmaktadır.

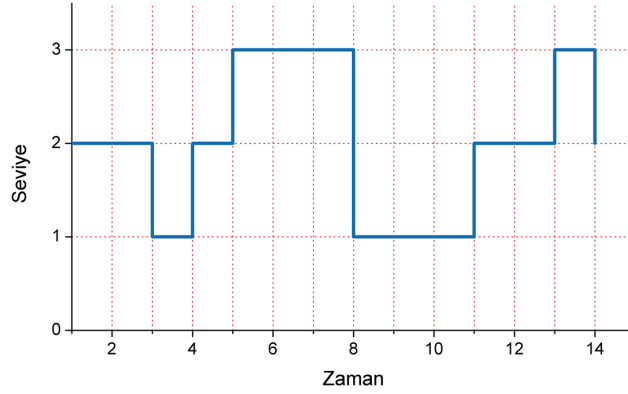
1. Örnekleme frekansı (f_s) ne kadardır?
2. Örnekleme periyodu (T_s) ne kadardır? Hesaplayınız.

İşaretin yalnızca belirli değerleri alması için değerler, seviyelere ayrılırsa bu işlem **kuantalama** olarak adlandırılır. Örneğin sıcaklığın izlendiği bir sistemde, sıcaklık değerlerine göre Tablo 4.2’de görülen şekilde bir seviyeleme yapılabilir. Kuantalama işleminde, okunan her sıcaklık ölçüm değeri, karşısında belirtilen seviye ile ifade edilir.

Tablo 4.2: Sıcaklık Seviyeleri

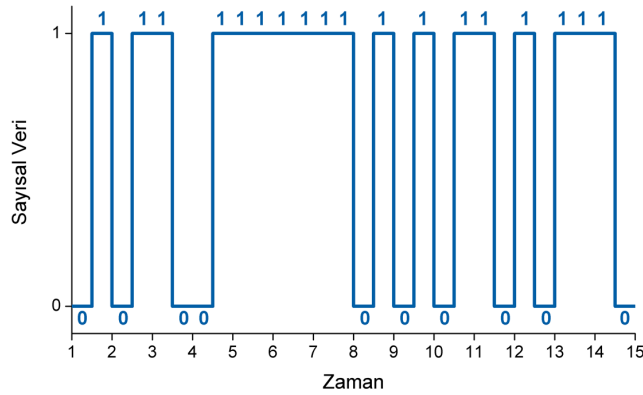
Aralık	Seviye	İkili Tabanda Sayı Karşılığı
Sıcaklık < -10 °C	0	00
-10 °C ≤ Sıcaklık < 0 °C	1	01
0 °C ≤ Sıcaklık < 10 °C	2	10
10 °C ≤ Sıcaklık	3	11

Bu seviyeleme, ihtiyaca ve sistem gereksinimlerine göre farklı şekillerde yapılabilir. Tablo 4.1’de verilen ölçüm değerlerinin Tablo 4.2’de verilen seviyelere göre kuantalama işlemi sonucunda Görsel 4.4’te verilen grafik elde edilir.



Görsel 4.4: Kuantalama yapılan işaret

Kuantalama yapılan işarete bulunan sayılar, ikilik tabanda sayı karşılıklarına (0 ve 1’ler kullanılarak) dönüştürülür, böylece **sayısal işaret** ya da diğer bir deyişle **dijital işaret** elde edilir. Bu işleme **sayısallaştırma** adı verilir. Sayısal işaretler, yalnızca **0** veya **1** değerlerini alan işaretlerdir. Bu nedenle sayısal işaretlerde yalnızca iki durum bulunur. Elektriksel işaret olarak genellikle 0 değeri **toprak**, 1 değeri ise genellikle **+3,3 V** ya da **+5 V** DC gerilim değerlerine karşılık gelir. Görsel 4.5’te bir sayısal işaret örneği verilmiştir.

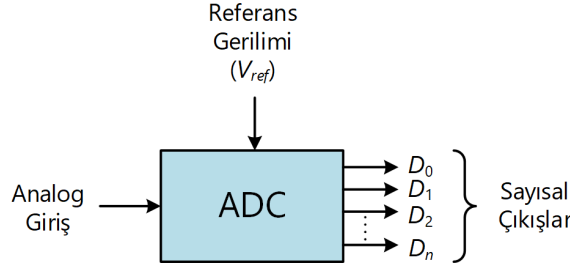


Görsel 4.5: Sayısal işaret

Tablo 4.2’de, sıcaklık örneğindeki her bir seviyenin ikilik tabanda sayı karşılıkları gösterilmiştir. Kuantalama işlemi sonucunda, ölçülen sıcaklık değerinin seviyesine karşılık gelen ikilik tabanda sayılar, farklı haberleşme protokollerine göre 0 veya 1 değerleri şeklinde mikrodenetleyici veya çevre birimlere iletilir.

4.1.2. ADC

Fiziksel ölçümler sonucunda elde edilen işaretler, genellikle analog işaretlerdir. Mesela ışık seviyesi fotodirençler yardımıyla gerilim değerlerine dönüştürülebilir. Bu işaretleri alarak sayısal aygıtlarda (örneğin mikro işlemci veya mikrodenetleyicilerde) işleyebilmek için sayısal kodlara çevirmek gerekir. Analog işaretleri sayısal çıkış kodlarına çevirmek için kullanılan aygıt **analog / sayısal çevirici [Analog to Digital Converter (ADC)]** adı verilir. Bir ADC'nin blok şeması Görsel 4.6'da verilmiştir.



Görsel 4.6: ADC blok şeması

Burada n indisi, sayısal çıkış kodunun bit sayısını ifade eder. Mikrodenetleyiciler 8, 10, 12 veya 16 bitlik sayısal çıkış kodları üretir. Atmega328P mikrodenetleyici, 10 bitlik sayısal çıkış kodu üretebilen bir ADC'ye sahiptir. Bit sayısı arttıkça ADC'nin üretebileceği sayı değeri artacağından analog / sayısal çevirme işleminin seviye sayısı artar. n değerine göre ADC sayısal çıkış kodunun alacağı en yüksek değer, aşağıdaki formül ile hesaplanabilir:

$$\text{En Yüksek ADC Sayısal Çıkış Değeri} = 2^n - 1$$

ADC sayısal çıkış kodunun alacağı en küçük değer ise 0'dır. Dolayısıyla analog işaret, ADC tarafından 2^n seviyede temsil edilerek sayısal çıkış kodu üretilir..

ADC, analog giriş işaretini, uygulanan referans gerilime (V_{ref}) göre bir sayısal koda çevirir.



ÖRNEK 2

10 bitlik sayısal çıkışa sahip bir ADC, hangi değer aralığında çıkış verebilir? Hesaplayınız.

Verilenler: Çıkış bit sayısı $n = 10$

İstenen : ADC çıkış kodu değer aralığı

ÇÖZÜM: Yukarıda verilen formüle göre 10 bitlik çıkışa sahip bir ADC, girişten uygulanan analog işaretin seviyesine göre 0 ile $(2^{10} - 1) = 1023$ değerleri arasında sayısal kodlar üretir. Verilen ADC için analog işaret, toplamda **1024** seviyede temsil edilir.

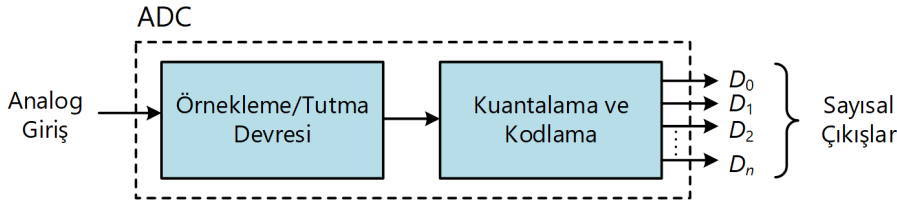


SIRA SİZDE

8 bitlik sayısal çıkışa sahip bir ADC'nin hangi değer aralığında sayısal çıkış verebileceğini hesaplayınız.

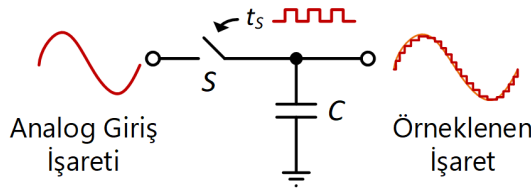
4.1.3. ADC'nin İç Yapısı

ADC, sürekli zamanlı işaretleri alarak ayrık zamanlı işaretlere dönüştürür. Sonrasında ayrık zamanlı işaretin ikilik tabandaki sayısal kod karşılığını çıkış olarak verir. Bir ADC iç yapısının blok şeması Görsel 4.7'de verilmiştir.



Görsel 4.7: ADC iç yapısının blok şeması

Örnekleme / Tutma [Sample/Hold (S/H)] Devresi: Girişe uygulanan analog işaretten örnek olarak alınan veriyi, çevirme işlemi gerçekleşinceye kadar, kısa süreli hafızada tutan devredir. Basit bir örnekleme / tutma devresi, Görsel 4.8'de verilmiştir. S anahtarının her bir t_s anında kapanması ile analog giriş işaretinin o andaki değeri, C kondansatöründe depolanarak saklanır. Kondansatör şarj olduğunda anahtar açılır. Böylece devrenin çıkışında örneklenen değer, bir sonraki örnekleme anına kadar hafızada tutulmuş olur.

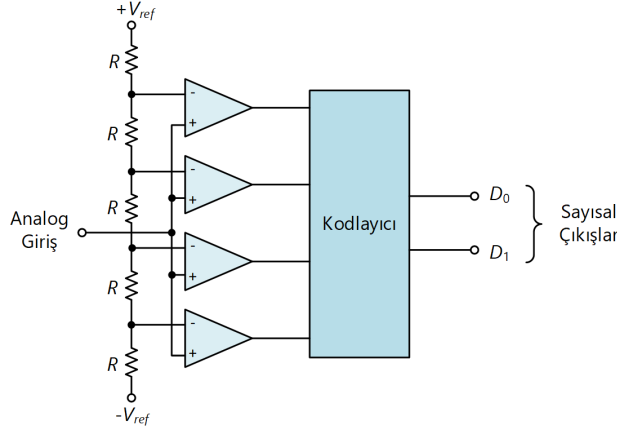


Görsel 4.8: Örnekleme / tutma devresi

Kuantalama ve Kodlama: Girişe uygulanan analog işaretten alınan örnekleri, kuantalama işlemine tabi tutarak seviyesini belirleyen ve sayısal çıkış koduna dönüştüren birimdir.

Örnek olarak Görsel 4.9'da 2 bitlik flash ADC'nin devre şeması görülmektedir. Devre şemasında; 4 adet karşılaştırmacı işlemsel yükselteç, 1 adet 4×2 kodlayıcı ve 5 adet gerilim bölücü direnç bulunmaktadır. $+V_{ref}$ ile $-V_{ref}$ gerilimleri arasındaki potansiyel fark, R dirençleri yardımıyla bölünerek her bir karşılaştırmacı, işlemsel yükseltecin eviren (-) girişlerine uygulanır.

Analog giriş işareti ise her bir işlemsel yükseltecin evirmeyen (+) girişine uygulanır. İşlemsel yükselteç, girişlerine verilen işaretleri karşılaştırır ve eğer uygulanan analog giriş işareti, referans gerilimden daha büyükse 1, diğer durumda 0 çıkışını verir. İşlemsel yükseltecin çıkışları, giriş öncelikli bir kodlayıcıya bağlıdır. Kodlayıcı, işlemsel yükselteçlerin çıkışına göre ikilik tabanda sayısal çıkış kodu üretir.



Görsel 4.9: Flash ADC devre şeması



SIRA SİZDE

Flash ADC dışında başka tipte ADC devreleri bulunup bulunmadığını genel ağ üzerinden araştırarak ADC devrelerinin kullanıldığı yerleri sınıfta arkadaşlarınızla paylaşınız.

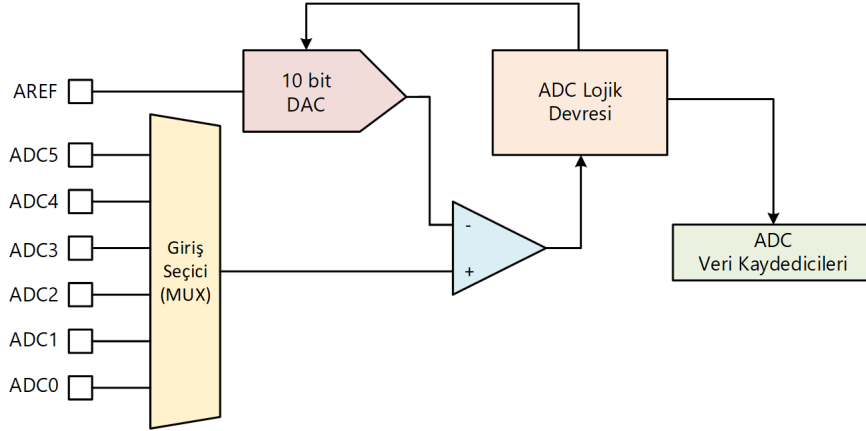
4.1.4. Mikrodenetleyici ADC Modülü

Birçok mikrodenetleyici, iç yapısında ADC modülüne sahiptir. Bu sayede haricî bir ADC entegresi kullanmaya gerek duyulmaz.

4.1.4.1. ADC Modülünün Yapısı

Mikrodenetleyiciler, bir veya birden çok analog giriş kanalından oluşan ADC modülü içerebilir. Örneğin Atmega328P mikrodenetleyici, DIP paketi için 6 adet analog giriş kanalından oluşan 10 bitlik bir ADC modülüne sahiptir. ADC0-ADC5 kanalları üzerinden analog işaret girişi yapılabilir, kanallardan uygulanan işaret sırasıyla çevirme işlemine tabi tutulur. ADC modülünün 10 bitlik bir kaydediciye (register) sahip olması uygulanan giriş işaretine göre 0 ile 1 023 arasında sayısal çıkış kodu değerlerinin üretilebileceğini ifade eder.

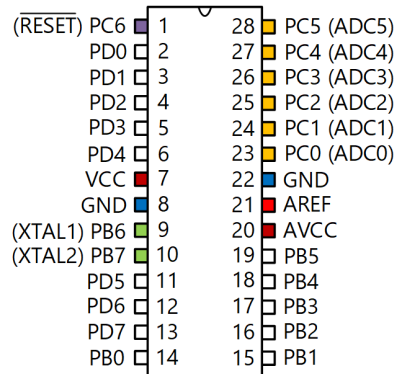
Mikrodenetleyici ADC modülünün basitleştirilmiş blok şeması, Görsel 4.10'da verilmiştir.



Görsel 4.10: Mikrodenetleyici ADC modülü blok şeması

Giriş seçici (MUX) tarafından seçilen ADC kanalına uygulanan işaret, MUX çıkışından karşılaştırıcı işlemsel yükseltecin (OP-AMP) evirmeyen (+) girişine uygulanır. OP-AMP'ın yükseltecin eviren (-) girişine ise ADC lojik devresi tarafından üretilen sayısal kodlar, 10 bitlik DAC tarafından analog işarete dönüştürülerek uygulanır. Uygulanan referans işaretler ile oluşan gerilim farkı kontrol edilir. Böylece giriş işareti, kuantalama işlemi gerçekleştirilerek sayısal çıkış koduna çevrilir ve çevrim sonucu ADC veri kaydedicilerine (ADCH ve ADCL) yazılır.

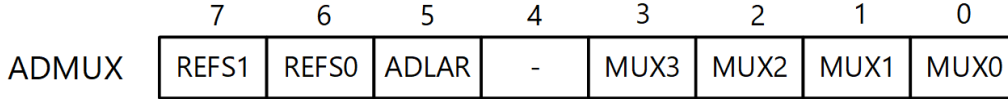
Atmega328P mikrodenetleyici entegresinin ADC modülü ile ilgili pinleri gösteren DIP soket pin şeması, Görsel 4.11'de verilmiştir. 23-28. pinler ADC kanallarıdır. ADC'nin çalışabilmesi için VCC ve AVCC pinlerine besleme gerilimi (3,3 V veya 5 V) bağlanmalıdır. GND ucu ise toprağa bağlanmalıdır. AREF ucuna ise ADC referans gerilimi olarak besleme gerilimi uygulanabilir. Ayrıca mikrodenetleyici resetleneceği zaman, RESET pinine lojik 0 uygulanır. Haricî osilatör gerekirse XTAL1 ve XTAL2 girişlerine bir kristal osilatör devresi bağlanabilir. Bu bölümde verilen örneklerde, mikrodenetleyici saat darbesi kaynağı olarak 1/8 ön ölçeklemeli dâhilî osilatör (1 MHz) kullanılacaktır.



Görsel 4.11: Atmega328P Mikrodenetleyicinin ADC modülü ile ilgili pinleri

4.1.4.2. Referans Gerilimi ve ADC Giriş Kanalı Seçimi

ADC çoğullayıcı seçim kaydedicisi (ADMUX) içerisinde, ADC'nin referans gerilimi ve giriş kanalını belirlemek için bitler yer alır. ADMUX kaydedicisinin bitleri Görsel 4.12'de verilmiştir.



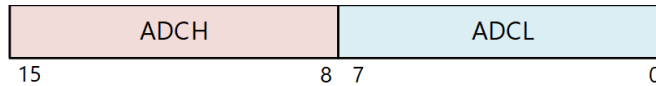
Görsel 4.12: ADMUX kaydedicisi

7. ve 6. Bitler REFS[1:0]: ADC'nin referans gerilimini belirler. ADC referans gerilimi (V_{REF}), ADC'nin çevirme gerilim aralığını belirler. AV_{CC} , ADC modülü için kullanılan besleme gerilimidir. V_{REF} , haricî AREF pininden uygulanan gerilim ya da AV_{CC} olarak belirlenebileceği gibi dâhilî 1,1 V referans gerilimi olarak da seçilebilir. AREF pinine haricî bir topraklanmış kondansatör eklenerek işaretteki gürültü önlenir. Eğer haricî bir referans gerilimi kullanılmayacaksa AV_{CC} veya dâhilî 1,1 V referans gerilimi seçeneklerinden biri seçilmelidir. V_{REF} seçim değerleri ve anlamları Tablo 4.3'te verilmiştir.

Tablo 4.3: Referans Gerilimi Seçimi

REFS1	REFS0	Referans Gerilimi Seçimi
0	0	AREF, Dâhilî V_{ref} kapalı
0	1	AREF pininde haricî kondansatörlü AV_{CC}
1	0	Rezerve
1	1	AREF pininde haricî kondansatörlü dâhilî 1,1 V referans gerilimi

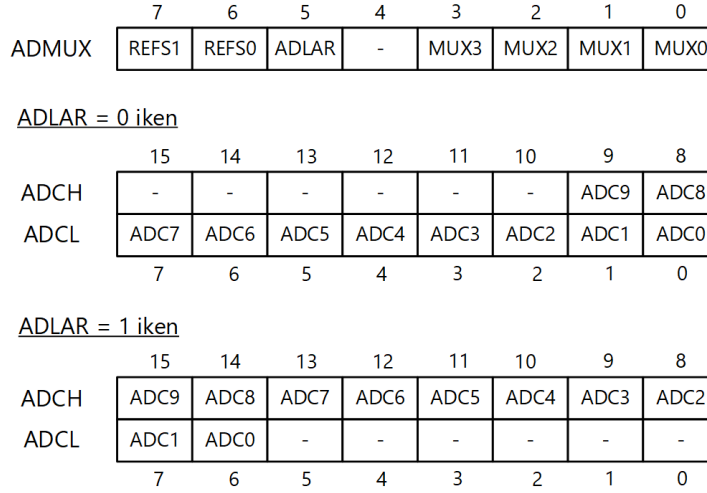
5. Bit ADLAR: Varsayılan olarak ADCH ve ADCL kaydedicilerine yazılan çevirme sonucu sağa yaslıdır, bu durum istenirse değiştirilebilir. Çevrim sonucunda, 0-7. bitler ADCL kaydedicisine, 8 ve 9. bitler ise ADCH kaydedicisine yazılmaktadır. ADCH kaydedicisinin kullanılmayan bitleri dikkate alınmaz. ADC veri kaydedicileri, Görsel 4.13'te gösterilmiştir.



Görsel 4.13: ADC veri kaydedicileri (ADCH ve ADCL)

ADC çevirme sonucu okuma işleminde öncelikle ADCL kaydedicisi, sonrasında ise ADCH kaydedicisi okunur. Bu okumalar esnasında, mikrodenetleyici tarafından ADC çevirme işlemi gerçekleşmez ve okuma işleminin tamamlanması beklenir.

ADLAR biti 1 olduğunda ADC veri kaydedicilerine yazılan sonucun sola yaslı olmasını sağlar. Bu durumda 2-9. bitler ADCH kaydedicisine, 0 ve 1. bit ise ADCL kaydedicisine yazılır. Bu durum, Görsel 4.14'te özetlenmiştir.



Görsel 4.14: ADLAR bitinin değerine göre ADCH ve ADCL kaydedicilerinin durumu

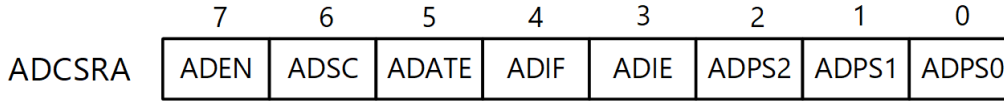
3. -0. Bitler MUX[3:0]: ADC giriş kanalını seçmek için ADC çoğullayıcı (multiplexer, MUX) seçim kaydedicisi (ADMUX) içerisinde bulunan MUX bitleri değiştirilir. Eğer çevirme işlemi sırasında bu bitler değiştirilirse çevirme işlemi bittikten sonra kanal değişikliği yapılır. Bu bitlerin anlamları Tablo 4.4'te verilmiştir.

Tablo 4.4: Giriş Kanalı Seçim Bitleri

MUX[3:0]	Tek Uçlu Giriş
0000	ADC0
0001	ADC1
0010	ADC2
0011	ADC3
0100	ADC4
0101	ADC5
1000	Sıcaklık Algılayıcı
1110	1,1 V
1111	0 V (Toprak)

4.1.4.3. ADC Modülünün Etkinleştirilmesi ve Çevirme İşlemine Başlama

ADC modülünün etkinleştirilmesi ve çevirme işlemine başlanması ADC kontrol ve durum kaydedicisi A (ADCSRA) üzerinden ayarlanır. ADCSRA kaydedicisinin bitleri Görsel 4.15'te verilmiştir.



Görsel 4.15: ADCSRA kaydedicisi

7. Bit ADEN: ADC modülü, ADCSRA kaydedicisinde yer alan ADC etkinleştirme (ADEN) bitinin 1 yapılması ile etkinleştirilir. ADEN biti 0 olduğunda ADC modülü kapanır ve güç tüketmez. Dolayısıyla ADC modülünün kullanılmadığı zamanlarda ADEN biti, 0 yapılmalıdır.

6. Bit ADSC: Çevirme işleminin başlaması için ADC çevirme işlemine başla (ADSC) biti 1 yapılır. Bu bit, çevirme işlemi süresince lojik 1 olarak kalır ve işlem tamamlandığında mikrodenetleyici tarafından sıfırlanır.

**NOT**

Çevirme işleminin başlatılması için güç kesme kaydedicisi [Power Reduction Register (PRR)] içerisinde yer alan PRADC adlı, ADC güç kesme bitinin sıfırlanması gerekir. Bu bitin varsayılan değeri 0'dır.

İlk ADC çevirme işlemi, ADC'nin kurulum işlemleri nedeniyle 25 ADC saat darbesi kadar sürdürülür. İlk çevirme işleminden sonraki çevirmeler, 13 ADC saat darbesinde yapılır.

5. Bit ADATE: ADC otomatik tetikleme etkinleştirme bitidir. Bir çevirme işlemi, serbest çalışma modu haricinde değişik kaynaklar yardımıyla otomatik tetiklenerek de başlatılabilir. Otomatik tetikleme, ADCSRA kaydedicisi içerisinde bulunan ADC otomatik tetikleme etkinleştirme biti (ADATE) lojik 1 yapılarak etkinleştirilir. Tetikleme kaynağı, ADC tetikleyici seçim bitleri (ADTS[2:0]) ile seçilir. Bu bitler, ADCSRB kaydedicisi içerisinde yer alır. Seçilebilecek otomatik tetikleme kaynakları, Tablo 4.5'te verilmiştir. Tetikleyici, pozitif kenarlı (0'dan 1'e geçen) bir işaret üretirse ADC ön ölçekleyici (prescaler) yeniden kurulur ve çevirme işlemi başlar.

Tablo 4.5: ADC Otomatik Tetikleyici Kaynağı Seçimi

ADTS2	ADTS1	ADTS0	Tetikleme Kaynağı
0	0	0	Serbest Çalışma Modu
0	0	1	Analog Karşılaştırıcı
0	1	0	Haricî Kesme İsteği 0
0	1	1	Zamanlayıcı/Sayıc0 Karşılaştırma Eşleşmesi A
1	0	0	Zamanlayıcı/Sayıc0 Taşma
1	0	1	Zamanlayıcı/Sayıc1 Karşılaştırma Eşleşmesi B
1	1	0	Zamanlayıcı/Sayıc1 Taşma
1	1	1	Zamanlayıcı/Sayıc1 Yakalama Olayı

4. Bit ADIF: Bir ADC çevirme işlemi tamamlandığında ve veri kaydedicileri güncellendiğinde, ADC kesme bayrağı (ADIF) lojik 1 olur. Sonrasında, eğer kesme etkinleştirilmişse program, ADC çevirme işlemi tamamlama kesmesine dallanır.

3. Bit ADIE: ADC kesme etkinleştirme bitidir. Bu bit ve SREG kaydedicisindeki I-biti 1 olduğunda ADC çevirme işlemi tamamlama kesmesi çalışır.

2.-0.Bitler ADPS[2:0]: ADC ön ölçekleyici seçim bitleridir. Ön ölçekleyici, ADC'nin örnekleme frekansını belirler. Örneğin ADPS[2:0]=000 durumunda, ADC örnekleme frekansı, mikrodenetleyici çalışma frekansının yarısı olarak belirlenir. ADPS[2:0] = 101 olduğunda ise ADC örnekleme frekansı, mikrodenetleyici çalışma frekansının 1 / 32'si kadar olur. Tablo 4.6'da ön ölçekleyici seçim bitleri ve karşılıkları verilmiştir.

Tablo 4.6: ADC Ön Ölçekleyici Seçimi

ADPS2	ADPS1	ADPS0	Bölme Faktörü
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

4.1.4.4. Kullanılmayan Sayısal Girişlerin Devre Dışı Bırakılması

Analog giriş yapılacak pinlerden aynı zamanda sayısal giriş yapılmayacak ise bu pinlerin sayısal giriş tamponlarını devre dışı bırakmak suretiyle güç tüketimini azaltmak için sayısal giriş devre dışı bırakma kaydedicisi (DIDR0) kullanılır. DIDR0 kaydedicisinin bitleri, Görsel 4.16'da verilmiştir.

	7	6	5	4	3	2	1	0
DIDR0	-	-	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D

Görsel 4.16: DIDR0 kaydedicisi

Örneğin, ADC'nin 2, 3 ve 4. kanalları yalnızca analog giriş için kullanılacaksa ADC2D = 1, ADC3D = 1 ve ADC4D = 1 yapılır.

4.1.4.5. ADC Çevirme Sonucunun Bulunması

Çevirme sonucu, çevirme işlemi tamamlandıktan sonra (ADIF = 1 iken), ADCH ve ADCL kaydedicilerinden okunabilir. C dilinde kod yazarken okuma işlemi, doğrudan **ADC** değişkeni

üzerinden yapılır. ADC çevirme sonucu aşağıda verilen formülle bulunabilir:

$$\text{ADC Çevirme Sonucu} = V_{IN} \times 1024 / V_{REF}$$

Burada, V_{IN} seçilen giriş pinindeki gerilim, V_{REF} ise referans gerilimidir.

4.1.4.6. ADC Çevirme İşleminin Adımları

ADC çevirme işlemi için programlama adımları aşağıda belirtilmiştir:

1. Referans gerilimi belirlenir. Bunun için ADMUX kaydedicisinin REFS[1:0] bitleri Tablo 4.3'e göre ayarlanır.

2. İsteğe bağlı olarak ADC çevirme işlemi sonucunun ADCH ve ADCL kaydedicilerinde hizalama şekli belirlenir. Bunun için ADMUX kaydedicisinin ADLAR biti kullanılır. Varsayılan olarak ADLAR = 0 olup çevirme işlemi sonucu sağa yaslıdır. C dilinde program yazarken ADC isimli değişken ile ADC çevirme sonucu direkt okunduğundan bu bitin ayarlanmasına ihtiyaç duyulmaz.

3. ADC giriş kanalı seçimi yapılır. Bunun için ADMUX kaydedicisinin MUX[3:0] bitleri Tablo 4.4'e göre ayarlama yapılır.

4. ADC etkinleştirilir. Bunun için ADCSRA kaydedicisinin ADEN biti 1 yapılır.

5. İsteğe bağlı olarak ADC otomatik tetikleme seçeneği belirlenir. Varsayılan olarak ADATE = 0 iken otomatik tetikleme devre dışıdır. Bu durumda, birden fazla ADC çevirme işlemi yapılacağı zaman, her bir işlem, ADC çevirme işlemine başla biti (ADSC) ile manuel olarak başlatılabilir. Otomatik tetikleme etkinleştirildiği takdirde (ADATE = 1 iken), ADCSRB kaydedicisinde yer alan ADTS[2:0] bitleri, Tablo 4.5'e göre ayarlanır.

6. Ön ölçekleyici (prescaler) ayarı yapılır. Bunun için ADCSRA kaydedicisinin ADPS[2:0] bitleri Tablo 4.6'ya göre ayarlanır.

7. ADC çevirme işlemi başlatılır. ADCSRA kaydedicisinde yer alan ADCSC biti 1 yapılır.

8. Okuma işlemi bitene kadar (ADCSC = 0 olana kadar) beklenir.

9. ADC çevirme sonucu okunur. Bu değer, ADCH ve ADCL kaydedicilerinden ayrı ayrı okunabileceği gibi C ile programlama yapılırken doğrudan **ADC** değişkeninin içeriği de okunabilir.



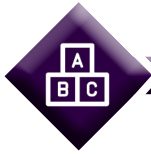
SIRA SİZDE

Bir ADC çevirme işleminde dâhilî 1,1 V referans gerilimi kullanılacaktır. ADCH ve ADCL değişkenlerine yazılacak değer sağa yaslı olacak ve ADC2 kanalı yalnızca analog giriş için kullanılacaktır. Ayrıca ADC çevirme tamamlama kesmesi ve ADC modülü etkinleştirilip çevirme işlemi başlatılacaktır.

Bu işlemler için Tablo 4.7’de verilen kaydedicilerin içeriği nasıl olmalıdır? Tablodaki boş hücreleri doldurunuz.

Tablo 4.7: ADC Kaydedicileri

Kaydedici	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADMUX	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0
				-				
ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
				0				
DIDR0	-	-	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
	-	-						



UYGULAMA

Adı:	LED’ler ile Gerilim Seviye Göstergesi	No : 4.1
AMAÇ:	ADC Modülü kullanarak LED’ler ile gerilim seviye göstergesi devresini tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda analog girişten 0-5 V arasında uygulanan gerilimi, ADC modülüyle çevirip B portuna bağlı 5 LED ile gerilim seviyesini gösteren bir devre oluşturunuz ve gömülü kodunu yazınız.

Analog giriş için ADC0 kanalını (PC0 pini), LED çıkışları için ise PB0-PB4 pinlerini kullanınız.

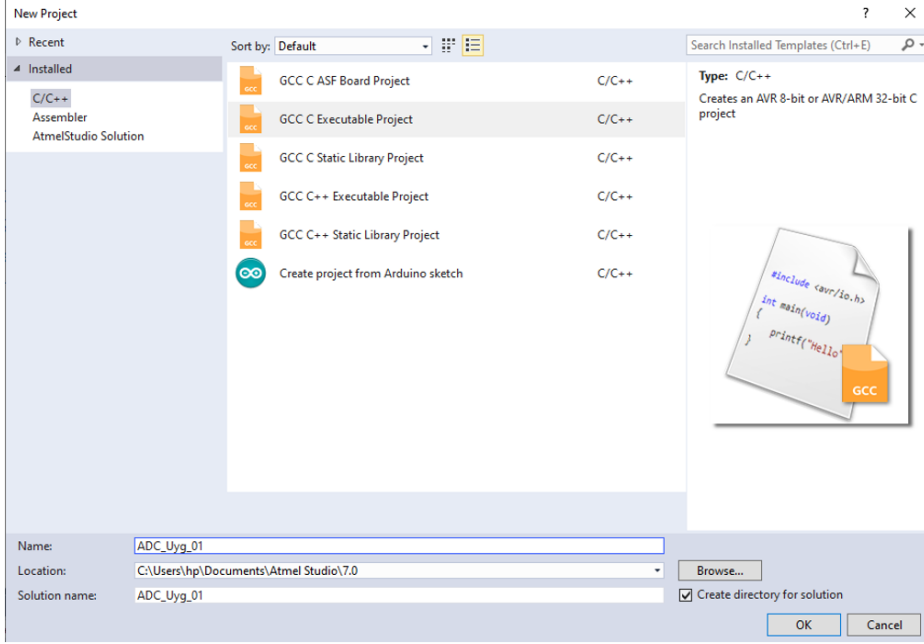
Kullanılacak Araç Gereç: Tablo 4.8.’de belirtilmiştir.

Tablo 4.8: Uygulama 4.1 için Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 adet
LED	5 mm, Kırmızı	5 adet
Direnç	180 Ω	5 adet
Direnç	4.7 k Ω	1 adet
Kondansatör	100 nF	1 adet
Potansiyometre	10 k Ω	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı teli		1 m
Yan keski		1 adet

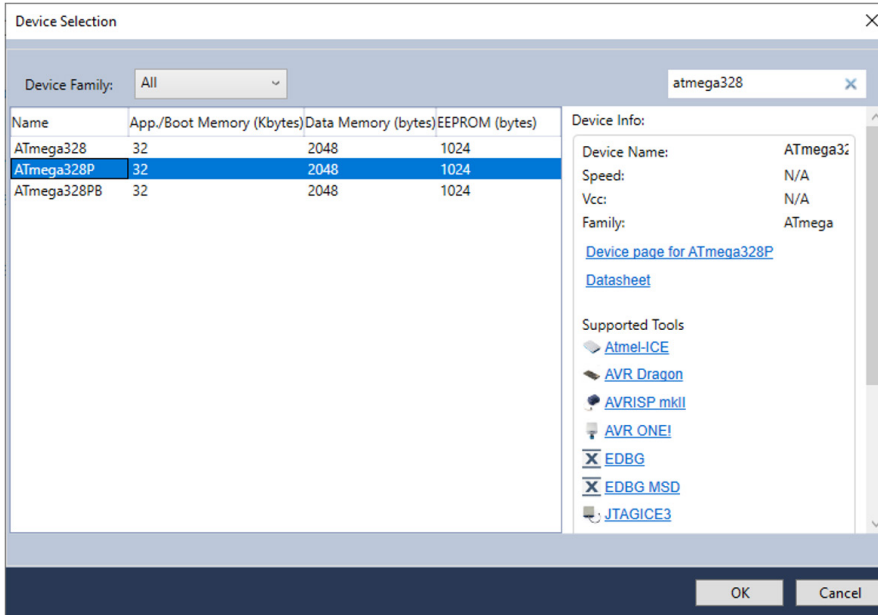
Uygulama Adımları:

1. Adım: Atmel Studio'da **File** menüsünden **New Project** seçeneğini seçiniz. Görsel 4.17'de görülen şekilde, **GCC C Executable Project** seçeneğini seçtikten sonra **Name** (proje adı) kısmına **ADC_Uyg_01** yazınız.



Görsel 4.17: New Project penceresi

2. Adım: Görsel 4.18'de görülen **Device Selection** penceresinden **Atmega328P** adlı mikrodenetleyiciyi seçiniz.



Görsel 4.18: Device Selection penceresi

3. Adım: main.c dosyası içerisine aşağıda verilen kodları yazınız.

```

#define F_CPU 1000000UL

#include <avr/io.h>
#include <util/delay.h>

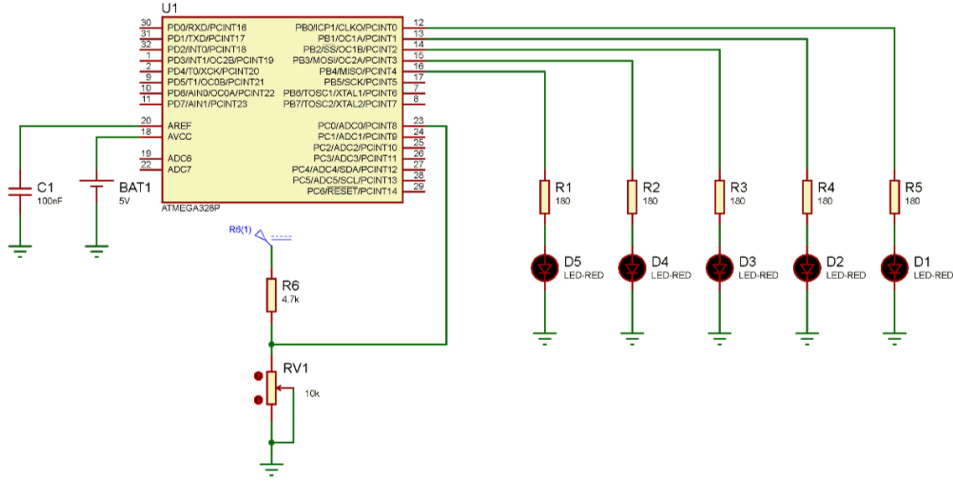
int main(void)
{
    volatile int cvalue;           // ADC çevirme sonucunun kaydedildiği değişken
    DDRB = 0xFF;                  // Port B'yi çıkış portu olarak ayarla.
    PORTB = 0x00;                 // Port B'yi sıfırla.
    ADMUX &= ~(1 << REFS1);       // Referans gerilimi seçimi:
    ADMUX |= (1 << REFS0);        // REFS[1:0]=01: AREF pininde haricî kondansatörlü AVCC
    ADMUX &= ~(1 << ADLAR);       // Sonuç hizalama (ADLAR=0: sağa yaslı)
    ADMUX &= ~((1 << MUX3) | (1 << MUX2) | (1 << MUX1) | (1 << MUX0));
                                // ADC giriş kanalı seçimi (MUX[3:0]=0000: ADC0 kanalı)
    ADCSRA |= (1 << ADEN);        // ADC etkin (ADEN=1).
    ADCSRA &= ~(1 << ADSC);       // Otomatik tetikleme devre dışı (ADSC=0).
    ADCSRA &= ~((1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0));
                                // Prescaler (Ön ölçekleyici) (ADPS[2:0]=000: /2)

    while (1)
    {
        ADCSRA |= (1 << ADSC);    // ADC çevirme işlemini başlat (ADSC=1).
        while(ADCSRA & (1 << ADSC)); // Okuma işlemi bitene kadar bekle (ADSC=0).
        cvalue = ADC;             // ADC değerini oku.
        if(cvalue < 170)          // ADC değeri 170'ten küçükse
            PORTB = 0b000000001; // 1. LED'i yak.
        else if((cvalue >= 170) && (cvalue < 340))
            PORTB = 0b000000011; // ADC değeri 170 ile 339 arasındaysa
                                // 1. ve 2. LED'leri yak.
        else if((cvalue >= 340) && (cvalue < 510))
            PORTB = 0b000000111; // ADC değeri 340 ile 509 arasındaysa
                                // 1-3. LED'leri yak.
        else if((cvalue >= 510) && (cvalue < 680))
            PORTB = 0b000001111; // ADC değeri 510 ile 679 arasındaysa
                                // 1-4. LED'leri yak.
        else if((cvalue >= 680) && (cvalue < 850))
            PORTB = 0b000011111; // ADC değeri 680 ile 849 arasındaysa
                                // 1-5. LED'leri yak.
        _delay_ms(500);           // 500 ms bekle.
    }
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğuna dikkat ediniz.

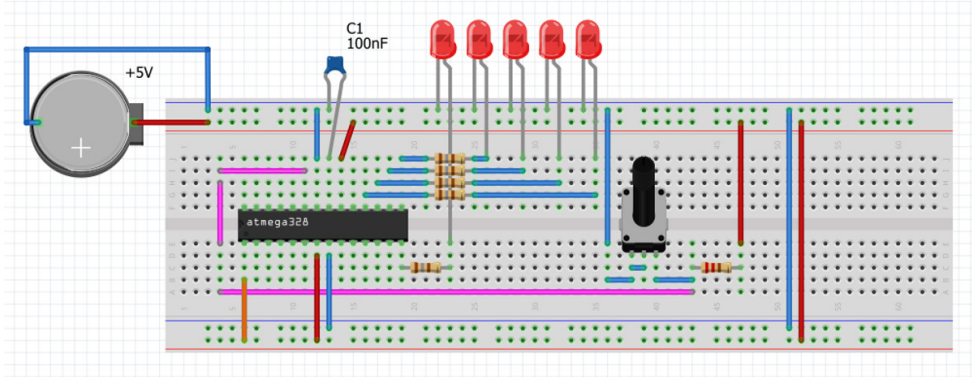
5. Adım: Devre simülasyon programında Görsel 4.19'da verilen devreyi kurarak simüle ediniz.



Görsel 4.19: Uygulama 4.1 için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.20’de verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.20: Uygulama 4.1 için breadboard üzerine kurulan devre

8. Adım: Potansiyometreyi değiştirerek LED’leri izleyiniz. Analog işaret girişini bir voltmetre yardımıyla ölçerek LED’lerin durumu ile karşılaştırınız. Karşılaştırma için Tablo 4.9’u kullanınız. Devrenin çalışmasını öğretmeninizle birlikte değerlendiriniz.

Tablo 4.9: Uygulama 4.1 için Karşılaştırma Tablosu

Yanan LED’ler	Ölçülen En Küçük Gerilim (V)	Ölçülen En Yüksek Gerilim (V)
Yalnızca 1. LED		
1 ve 2. LED’ler		
1, 2 ve 3. LED’ler		
1, 2, 3 ve 4. LED’ler		
Tüm LED’ler		

**NOT**

Kullandığınız potansiyometre, logaritmik olarak değişmekteyse gerilim ayarlama sırasında LED'lerin değişimi farklı sürelerde olabilir.

9. Adım: Güç kaynağını kapatınız.

10. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Potansiyometre değiştirildiğinde LED'lerin sırayla artıp azalarak yandığı gözlemlendi.		
6. Analog giriş işareti voltmetre ile ölçülerek LED'lerin durumu doğrulandı.		
7. Temizliğe dikkat edildi.		

**SIRA SİZDE**

4.1 No.lu uygulamada 5 adet LED kullanılarak gerilim seviyesi ölçülmüştür. Yalnızca 4 adet LED kullanarak 0-5 V arasında seviye ölçümü yapmak için yazılım nasıl değiştirilebilir? Yazılımı kodlayarak simüle ediniz.



UYGULAMA

Adı:	ADC Modülü ile Gerilim Ölçümü	No : 4.2
AMAÇ:	ADC Modülü kullanarak gerilim ölçümü yapan devreyi tasarlamak ve gömülü kodunu yazmak.	Süre: 60 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, mikrodenetleyici analog girişinden 0-5 V arasında uygulanan gerilimi, ADC modülü ile çevirip bir LCD panelde, ADC çevirme işlemi sonucunu ve gerilim değerini görüntüleyiniz.

Kullanılacak Araç Gereç: Tablo 4.10'da belirtilmiştir.

Tablo 4.10: Uygulama 4.2 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 adet
LCD	16x2 karakter	1 adet
Direnç	1 kΩ	2 adet
Kondansatör	100 nF	1 adet
Potansiyometre	5 kΩ	2 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı teli		1 m
Yan keski		1 adet

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **ADC_Uyg_02** adlı yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: Projenizde **main.c** dosyasının bulunduğu dizinde **Include** adında bir dizin oluşturup içerisine **lcd.h** adında bir dosya açarak içerisine aşağıda verilen kodları yazınız. Bu kodlar LCD'yi kullanabilmeniz için gereklidir.

```

/*
4-Bit 16x2 LCD Örneği
*/
#define F_CPU 1000000UL

#include <avr/io.h>
#include <util/delay.h>

#define LCD_Port PORTD           // LCD portunu tanımla (PORTD).
#define LCD_DPin DDRD           // 4-bit pinleri tanımla (PORTD4-PORTD7).
#define RSPIN PDO               // RS pinini tanımla.
#define ENPIN PD1              // E pinini tanımla.

int runtime;                    // LCD için zamanlayıcı

void LCD_Action( unsigned char cmnd ) // LCD çalışma kodları
{
    LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0);
    LCD_Port &= ~(1 << RSPIN);
    LCD_Port |= (1 << ENPIN);
    _delay_us(1);
    LCD_Port &= ~(1 << ENPIN);
    _delay_us(200);
    LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);
    LCD_Port |= (1 << ENPIN);
    _delay_us(1);
    LCD_Port &= ~(1 << ENPIN);
    _delay_ms(2);
}

void LCD_Init (void)
{
    LCD_DPin = 0xFF;           // LCD pinlerini kontrol et (PORTD4-PORTD7).
    _delay_ms(15);             // LCD etkinleştirilinceye kadar bekle.
    LCD_Action(0x02);          // 4-bit kontrol
    LCD_Action(0x28);          // Kontrol matris ayarı
    LCD_Action(0x0c);          // İmleç devre dışı.
    LCD_Action(0x06);          // İmleci taşı.
    LCD_Action(0x01);          // LCD'yi temizle.
    _delay_ms(2);              // 2 ms bekle.
}

void LCD_Clear()               // LCD temizleme
{
    LCD_Action (0x01);          // LCD'yi temizle.
    _delay_ms(2);              // LCD temizlenene kadar bekle.
    LCD_Action (0x80);          // İmleci 1. satır 1. sütun pozisyonuna getir.
}

```

```

void LCD_Print (char *str)                // Karakter dizisini LCD'ye yaz.
{
    int i;
    for(i=0; str[i] != 0; i++)
    {
        LCD_Port = (LCD_Port & 0x0F) | (str[i] & 0xF0);
        LCD_Port |= (1 << RSPIN);
        LCD_Port |= (1 << ENPIN);
        _delay_us(1);
        LCD_Port &= ~(1 << ENPIN);
        _delay_us(200);
        LCD_Port = (LCD_Port & 0x0F) | (str[i] << 4);
        LCD_Port |= (1 << ENPIN);
        _delay_us(1);
        LCD_Port &= ~(1 << ENPIN);
        _delay_ms(2);
    }
}

void LCD_Printpos (char row, char pos, char *str) // Belirtilen lokasyona yaz.
{
    if (row == 0 && pos < 16)
        LCD_Action((pos & 0x0F) | 0x80);
    else if (row == 1 && pos < 16)
        LCD_Action((pos & 0x0F) | 0xC0);
    LCD_Print(str);
}

```

4. Adım: main.c dosyası içerisine aşağıda verilen kodları yazınız.

```

#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include ".\Include\lcd.h"                // lcd.h dosyasını dâhil et

int main(void)
{
    volatile int cvalue;                  // ADC çevirme sonucunun kaydedildiği değişken
    volatile float vvalue;               // Gerilim değerinin kaydedildiği değişken
    char showcvalue [16];                // ADC çevirme sonucu karakter dizisi
    char showvvalue [16];                // Gerilim değeri karakter dizisi
    LCD_Init();                          // LCD'yi tanı.
    ADMUX &= ~(1 << REFS1);               // Referans gerilimi seçimi: REFS[1:0]=01:
    ADMUX |= (1 << REFS0);                // AREF pininde haricî kondansatörlü AVCC
    ADMUX &= ~(1 << ADLAR);               // Sonuç hizalama (ADLAR=0: sağa yaslı)
    ADMUX &= ~((1 << MUX3) | (1 << MUX2) | (1 << MUX1) | (1 << MUX0));
                                           // ADC giriş kanalı seçimi (MUX[3:0]=0000: ADC0)
    ADCSRA |= (1 << ADEN);                // ADC etkin (ADEN=1).
    ADCSRA &= ~(1 << ADSC);               // Otomatik tetikleme devre dışı (ADSC=0).
    ADCSRA &= ~((1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0));
                                           // Prescaler (Ön ölçekleyici) (ADPS[2:0]=000: /2)
}

```

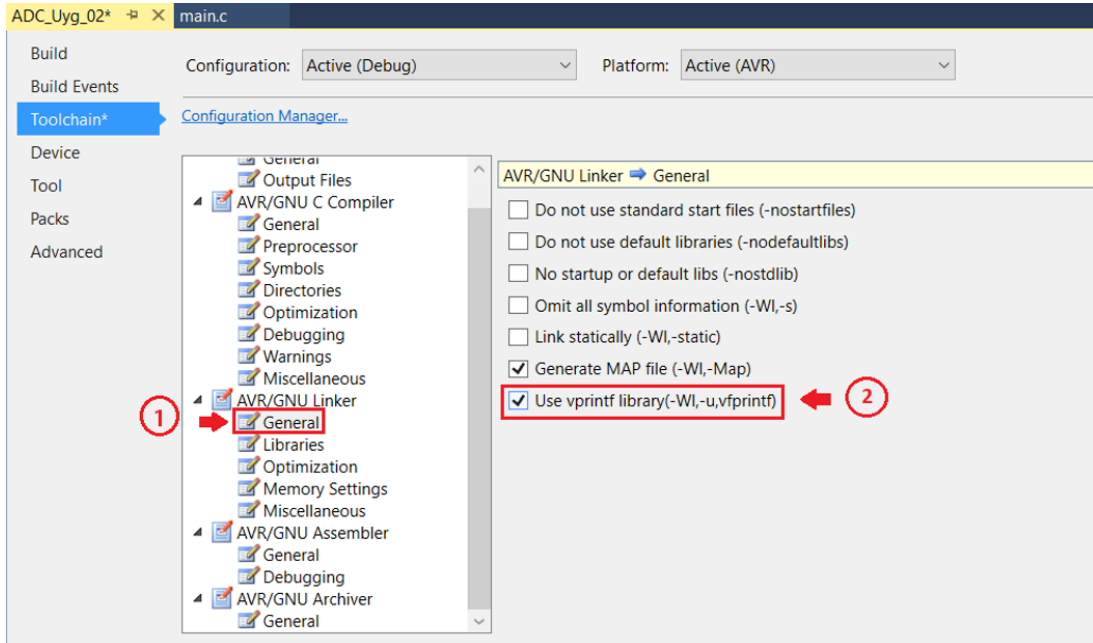
```

while (1)
{
    ADCSRA |= (1 << ADSC); // ADC çevirme işlemini başlat (ADSC=1).
    while(ADCSRA & (1 << ADSC)); // Okuma işlemi bitene kadar bekle (ADSC=0).
    cvalue = ADC; // ADC değerini oku.
    vvalue = (float)cvalue * 5 / 1024.0; // Gerilim değerini hesapla.
    itoa (cvalue,showcvalue,10); // cvalue sayısını karakter dizisi yap.
    sprintf (showvvalue,"%2.2f V",vvalue); // vvalue sayısını karakter dizisi yap.

    LCD_Clear(); // LCD'yi temizle.
    LCD_Print("ADC : "); // "ADC: " yaz.
    LCD_Print(showcvalue); // ADC değerini yaz.
    LCD_Action(0xC0); // İkinci satır, birinci sütuna git.
    LCD_Print("Gerilim:"); // "Gerilim: " yaz.
    LCD_Print(showvvalue); // Gerilim değerini yaz.
    _delay_ms(1 000); // 1000 ms bekle.
}
}

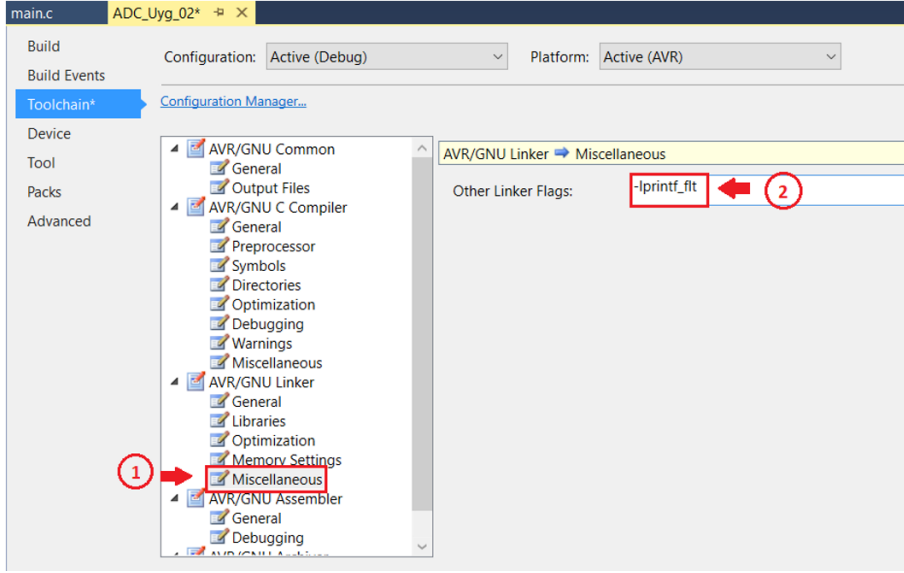
```

5. Adım: Kodlar içerisinde kullandığınız *sprintf()* fonksiyonunun doğru çalışabilmesi için **Debug** menüsünden **<Projenizin adı> Properties...** seçeneğini seçiniz. Görsel 4.21’de gösterildiği üzere **Toolchain** ana sekmesinde yer alan **AVR/GNU Linker** başlığı altında **General** sekmesine girerek “Use vprint library(-Wl,-u,vprintf)” seçeneğini işaretleyiniz.



Görsel 4.21: Project Properties penceresi

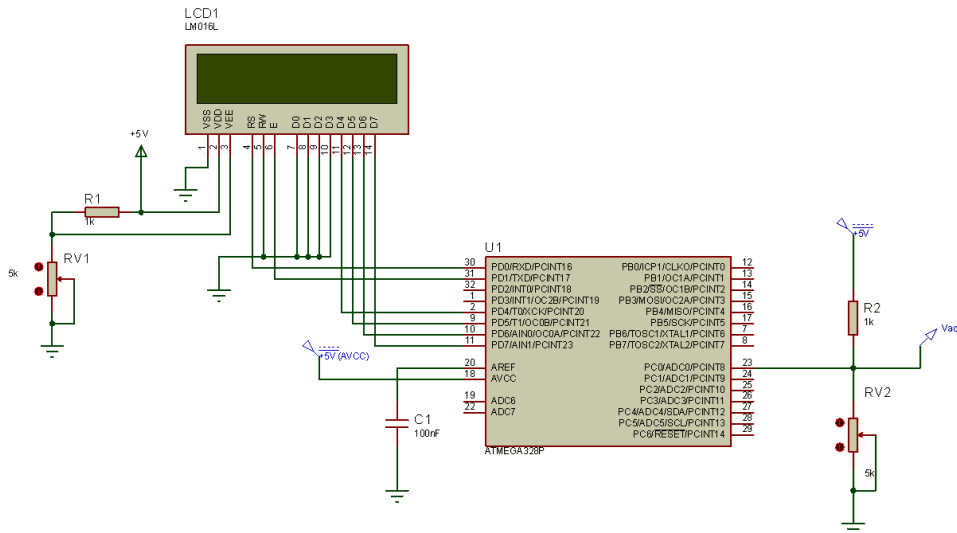
6. Adım: Görsel 4.22’de gösterilen pencerede, **Toolchain** sekmesinde **AVR/GNU Linker** başlığı altında **Miscellaneous** sekmesini tıklayınız. Burada **Other Linker Flags** alanına “-lprintf_flt” yazınız.



Görsel 4.22: Project Properties penceresi

7. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğuna dikkat ediniz.

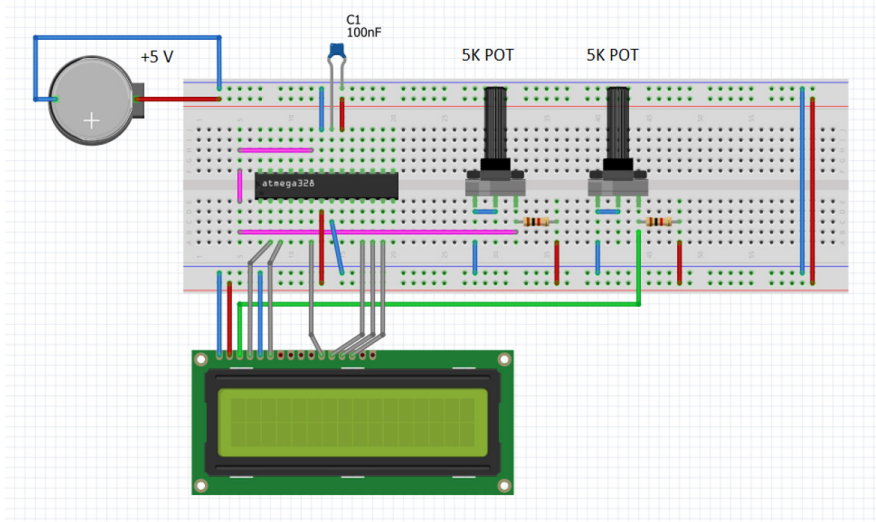
8. Adım: Devre simülasyon programında Görsel 4.23’te verilen devreyi kurarak simüle ediniz.



Görsel 4.23: Uygulama 4.2 ve 4.3 için simülasyon devresi

9. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

10. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.24'te verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.24: Uygulama 4.2 ve 4.3 için breadboard üzerine kurulan devre

11. Adım: Potansiyometreyi değiştirerek LCD üzerinde yazan değerleri izleyiniz. Analog işaret girişini bir voltmetre yardımıyla ölçerek LCD üzerinde okunan değer ile karşılaştırınız. Karşılaştırma için Tablo 4.11'i kullanınız. Devrenin çalışmasını öğretmeninizle birlikte değerlendiriniz.

Tablo 4.11: Uygulama 4.2 için Ölçüm Sonuçları

Ölçüm No	LCD Üzerinde Okunan Değer	ADC Değeri	Analog Girişte Ölçülen Gerilim (V)
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			

12. Adım: Güç kaynağını kapatınız.

13. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Potansiyometre ayarlandığında LCD üzerindeki değerler değişti.		
6. Analog giriş işareti voltmetre ile ölçülüp LCD üzerindeki sonuçlar doğrulandı.		
7. Temizliğe dikkat edildi.		



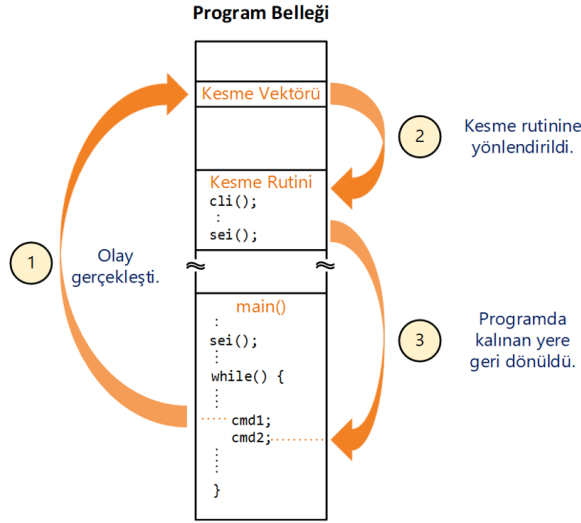
SIRA SİZDE

4.2 No.lu uygulamada, referans gerilimi değiştiğinde sonuçlar nasıl değişir? Referans gerilimini değiştirip deneyerek sonucu sınıfta arkadaşlarınızla birlikte yorumlayınız.

4.1.4.7. ADC Çevirme İşlemi Tamamlama Kesmesi Kullanma

Kesme (interrupt); mikrodenetleyicide bir olayın gerçekleşmesinin donanım tarafından takip edilmesi, olayın gerçekleşmesi hâlinde ilgili bayrak bitinin 1 olması ve programın bellekteki ilgili kesme kodlarına dallanarak bu kodları icra etmesidir. Örneğin ADC modülü için ADC çevirme işlemi sırasında ADCSRA kaydedicisinde yer alan ADSC biti 1 iken ADC çevirme işlemi tamamlandığında bu bit sıfırlanır ve ADIF kesme bayrağı 1 olur. Bu durumda, eğer kesmeler etkinse program ilgili kesme vektörünün olduğu adrese dallanır. Kesme vektöründe, ilgili kesme rutininin adresi bulunur. Program, kesme vektöründen ilgili kesme rutinine yönlendirilir. Kesme rutinindeki komutlar icra edildikten sonra, ADIF kesme bayrağı sıfırlanır,

program kaldığı yere geri döner ve çalışmaya devam eder. Kesme işleminin adımları, Görsel 4.25'te gösterilmiştir.



Görsel 4.25: Kesme işleminin adımları

4.1 ve 4.2 No.lu uygulamalarda verilen gömülü kodlarda, ADC çevirme işleminin tamamlanıp tamamlandığını tespit etmek amacıyla aşağıdaki kod kullanılmıştır:

```
while(ADCSRA & (1 << ADSC));           // Okuma işlemi bitene kadar bekle (ADSC=0).
```

Yukarıda verilen kod ile ADCSRA kaydedicisinin ADSC biti sürekli kontrol edilir, ADSC = 1 iken parantez içindeki koşul doğru olduğundan döngü devam eder ancak ADSC = 0 olduğunda, **while** döngüsünden çıkılarak ardından gelen kodlar çalışmaya devam eder. Dikkat edilirse **while** döngüsü içerisindeyken, mikrodenetleyici meşgul olup başka bir işlem yapma olanağı yoktur. Diğer bir ifadeyle ADC çevirme işlemi tamamlayıncaya kadar geçen sürede mikrodenetleyici boştaadır. Mikrodenetleyicinin daha etkin çalışabilmesi için kesmelerden yararlanılır.

Kesmeleri kullanmak için ilgili modülün kesmesi ile global kesmeleri etkinleştirmek ve kesme vektörünün altına olay gerçekleşince yapılacak işlemlerin kodunu tanımlamak gereklidir. Örneğin ADC modülü için ADC çevirme işlemi tamamlama kesmesi, ADSRA kaydedicisinin içerisinde yer alan ADC kesme etkinleştirme (ADIE) bitinin 1 yapılması ile etkin hâle gelir. Sonrasında *sei()* komutu ile mikrodenetleyici kesmeleri (SREG kaydedicisi 1 biti) etkinleştirilir. Bu durumda, program, ADC çevirme işlemini tamamladığında doğrudan ADC çevirme işlemi tamamlama kesme vektörüne dallanacağı için artık **while** ile ADSC bitini sürekli kontrol etmeye gerek duyulmaz. Bu işlem, mikrodenetleyici donanımı tarafından takip edilir. Kesme vektörüne gidildiğinde, ADIF kesme bayrak biti otomatik olarak temizlenir.

Kesme rutininin içerisinde kesintiye uğramaması gereken (kritik) işlemler yapılırken, öncelikle C dilinde *cli()* komutu ile global kesmeler devre dışı yapılır. Çünkü kesme rutini icra edilirken başka bir kesme işleminin gerçekleşmesi istenmez. Kesme rutini tamamlandığında *sei()* komutu ile tekrar global kesmeler etkinleştirilir.



UYGULAMA

Adı:	ADC Çevirme İşlemi Tamamlama Kesmesi Kullanma	No : 4.3
AMAÇ:	ADC Modülü ve kesme özelliğini kullanarak gerilim ölçümü yapan devreyi tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda 4.2 No.lu uygulamada verilen görevi, ADC çevirme işlemi tamamlama kesmesi kullanarak yapınız.

Analog girişten 0-5 V arasında uygulanan gerilimi ADC ile çevirerek LCD panelde, ADC çevirme işlemi sonucunu ve gerilim değerini görüntüleyiniz.

Kullanılacak Araç Gereç: Tablo 4.12’de belirtilmiştir.

Tablo 4.12: Uygulama 4.3 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 adet
LCD	16x2 karakter	1 adet
Direnç	1 kΩ	2 adet
Kondansatör	100 nF	1 adet
Potansiyometre	5 kΩ	2 adet
Breadboard		1 adet
DC Güç Kaynağı	5 V	1 adet
Mikrodenetleyici Programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı teli		1 m
Yan keski		1 adet

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **ADC_Uyg_03** adlı, yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: Projenizde **main.c** dosyasının bulunduğu dizinde **Include** adında bir dizin oluşturup içerisine 4.1 No.lu uygulamanın 3. adımında verilen **lcd.h** adındaki dosyayı ekleyiniz. Bu başlık dosyası LCD’yi kullanabilmeniz için gereklidir.

4. Adım: main.c dosyası içerisinde aşağıda verilen program kodlarını yazınız.

```

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <stdlib.h>
#include "..\Include\lcd.h"

volatile int cvalue;           // ADC çevirme sonucunun kaydedildiği değişken
volatile float vvalue;        // Gerilim değerinin kaydedildiği değişken
char showcvalue [16];         // ADC çevirme sonucu karakter dizisi
char showvvalue [16];         // Gerilim değeri karakter dizisi

ISR(ADC_vect)                 // ADC çevirme işlemi tamamlama kesme rutini
{
    cli();                    // Kesmeler devre dışı.
    cvalue = ADC;             // ADC değerini oku.
    vvalue = (float)cvalue * 5 / 1024.0;

                                // Gerilim değerini hesapla.
    itoa (cvalue, showcvalue, 10); // cvalue sayısını karakter dizisi yap.
    sprintf (showvvalue, "%2.2f V", vvalue); // vvalue sayısını karakter dizisi yap.
    LCD_Clear();               // LCD'yi temizle.
    LCD_Print("ADC : ");       // "ADC: " yaz.
    LCD_Print(showcvalue);      // ADC değerini yaz.
    LCD_Action(0xC0);           // İkinci satır, birinci sütuna git.
    LCD_Print("Gerilim: ");     // "Gerilim: " yaz.
    LCD_Print(showvvalue);      // Gerilim değerini yaz.
    sei();                     // Kesmeler etkin.
}

int main(void)
{
    LCD_Init();                // LCD'yi tanıt.
    ADMUX &= ~(1 << REFS1);     // Referans gerilimi seçimi: REFS[1:0]=01:
    ADMUX |= (1 << REFS0);       // AREF pininde haricî kondansatörlü AVCC
    ADMUX &= ~(1 << ADLAR);      // Sonuç hizalama (ADLAR=0: sağa yaslı)
    ADMUX &= ~((1 << MUX3) | (1 << MUX2) | (1 << MUX1) | (1 << MUX0)); // ADC giriş kanalı seçimi (MUX[3:0]=0000: ADC0)
                                // ADC etkin (ADEN=1).
    ADCSRA |= (1 << ADEN);       // Otomatik tetikleme devre dışı (ADATE=0).
    ADCSRA &= ~(1 << ADATE);      // Prescaler (Ön ölçekleyici) (ADPS[2:0]=000: /2)
    ADCSRA &= ~((1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0)); // ADC kesmesi etkin.
                                // Kesmeler etkin.
    sei();

    while (1)
    {
        ADCSRA |= (1 << ADSC);    // ADC çevirme işlemini başlat (ADSC=1).
        /* İstenirse burada başka işlemler de yapılabilir. */
        _delay_ms(500);           // 500 ms bekle.
    }
    return 0;
}

```

5. Adım: Kodlar içerisinde kullandığımız *sprintf()* fonksiyonunun doğru çalışabilmesi için 4.2 Nolu uygulamanın 5 ve 6. adımlarında verilen ayarlamaları yapınız.

6. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğuna dikkat ediniz.

7. Adım: Devre simülasyon programında, Görsel 4.24'te verilen devreyi kurarak simüle ediniz. Mikrodenetleyici yazılımı olarak bu uygulamada oluşturduğunuz HEX dosyasını tanımlayınız.

8. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

9. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine, Görsel 4.25'te verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.

10. Adım: Potansiyometreyi değiştirerek LCD üzerinde yazan değerleri izleyiniz. Devrenin çalışmasını öğretmeninizle birlikte değerlendiriniz.

11. Adım: Güç kaynağını kapatınız.

12. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Potansiyometre ayarlandığında LCD üzerindeki değerler değişti.		
6. Analog giriş işareti voltmetre ile ölçülüp LCD üzerindeki sonuçlar doğrulandı.		
7. Temizliğe dikkat edildi.		

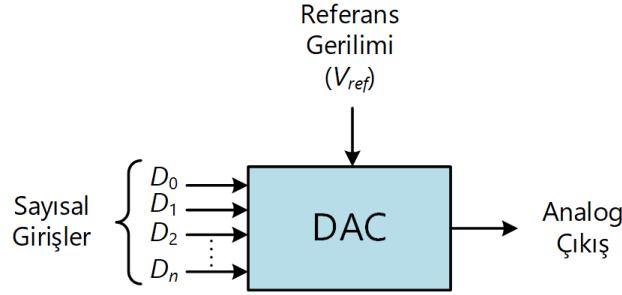


SIRA SİZDE

4.3 No.lu uygulamada verilen çalışmada, ADC okuma işlemleri arasında başka komutlar çalıştırınız. Örneğin sayısal çıkış PBO pinine bağlanan bir LED'in 500 ms aralıklarla yanıp sönmesini sağlayınız (**İpucu:** Bekleme için *delay_ms()* komutunu kullanabilirsiniz.).

4.1.5. DAC

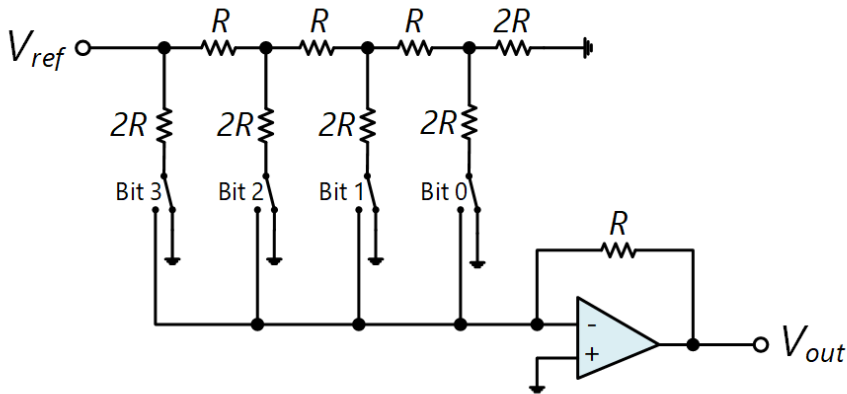
Sayısal / analog çeviriciler [Digital to Analog Converter (DAC)], girişinden uygulanan sayısal kodu, çıkışında analog işarete (gerilim veya akıma) çevirir. ADC tarafından gerçekleştirilen işlemin tersini yapar. Bir DAC elemanının blok şeması, Görsel 4.26'da verilmiştir.



Görsel 4.26: Sayısal / analog çevirici blok şeması

Örnek olarak Görsel 4.27'de 4 bitlik R - $2R$ merdiven direnç tipi DAC devre şeması verilmiştir. Anahtarlar, sayısal giriş kodunu temsil etmektedir. Uygulanan referans gerilim (V_{ref}), gerilim bölücü dirençler yardımıyla her bir anahtar üzerine kademeli olarak düşürülür. Böylece Bit 3'ten Bit 0'a doğru her bir anahtara uygulanan gerilim, bir sonraki anahtara uygulanan gerilimin iki katı kadardır. Bit 3 anahtarına V_{ref} , Bit 2 anahtarına $V_{ref}/2$, Bit 1 anahtarına $V_{ref}/4$ ve Bit 0 anahtarına $V_{ref}/8$ gerilimleri uygulanır.

Örneğin 0101 sayısal giriş kodu için Bit 2 ve Bit 0 anahtarlarını kapalı konuma getirdiğimizde $V_{ref}/2$ ve $V_{ref}/8$ gerilimleri, eviren yükseltecin girişine uygulanarak çıkış gerilimini (V_{out}) belirler. Çıkış gerilimi kazancı, eviren yükseltecin R direncinin değerinin değiştirilmesi ile artırılıp azaltılabilmektedir.

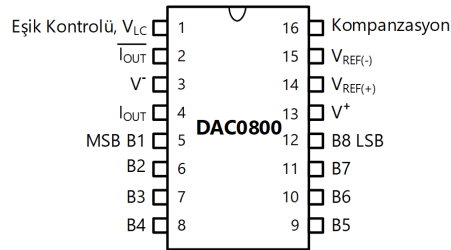


Görsel 4.27: R - $2R$ merdiven direnç tipi DAC devre şeması

R-2R merdiven direnç tipi dışında başka tipte DAC devreleri bulunup bulunmadığını genel ağ üzerinden araştırarak DAC devrelerinin kullanıldığı yerleri sınıfta arkadaşlarınızla paylaşınız.

4.1.5.1. Haricî DAC Entegresi Kullanma

Atmega328P mikrodenetleyicisi DAC modülü içermediğinden haricî bir DAC entegresi kullanarak DAC işlemleri gerçekleştirilebilir. Örneğin DAC0800 entegresi, 8 bitlik yüksek hızlı ve akım çıkışlı bir DAC entegresidir. 8 bitlik paralel girişe sahip olup çıkış ve tümleyen çıkış olmak üzere iki çıkışı bulunmaktadır. DAC0800 entegresinin pin şeması Görsel 4.28’de verilmiştir.

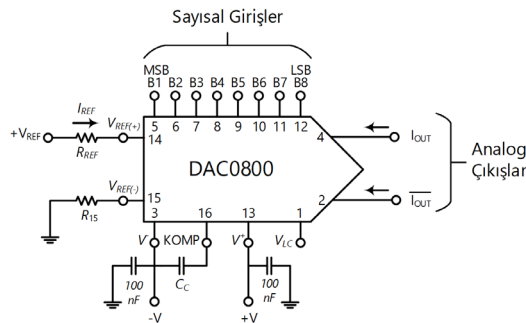


Görsel 4.28: DAC0800 entegresi DIP soket pin şeması

DAC0800, 16 pine sahiptir. V^+ ve V^- pinleri çalışma gerilimleri olup B_1 en yüksek değerli bit (MSB), B_8 ise en düşük bittir (LSB). $V_{REF(+)}$ ve $V_{REF(-)}$ girişlerine verilen işaret, çıkış akımını belirler. Çıkış akımı (I_{OUT}) aşağıdaki eşitlikten elde edilebilir:

$$I_{OUT} = \frac{V_{REF(+)}}{R_{REF}} \cdot \frac{B}{256}$$

Burada B , B_1 - B_8 paralel veri girişinden girilen değeri ifade eder. R_{REF} ise $V_{REF(+)}$ pinine bağlı direnç değeridir. DAC0800 entegresinin DAC işlemi için örnek uygulaması Görsel 4.29'da verilmiştir.



Görsel 4.29: DAC0800 entegrasi örnek uygulaması



SIRA SİZDE

DAC0800 entegresine $V_{REF(+)} = 5\text{ V}$ ve $V_{REF(-)} = 0\text{ V}$ (toprak) gerilimleri uygulanmış ve $R_{REF} = 4.7\text{ k}\Omega$ referans direnci bağlanmıştır. Gerekli besleme gerilimi ve sayısal girişten (B1-B8) 00110011 değeri uygulandığında oluşan çıkış akımını hesaplayınız.



UYGULAMA

Adı:	DAC Entegresi Kullanarak LED Parlaklığını Değiştirme	No : 4.4
AMAÇ:	DAC entegresi kullanarak LED parlaklığını değiştiren devreyi tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda bir DAC entegresi kullanarak LED parlaklığını değiştiren devreyi tasarlayınız ve gömülü kodunu yazınız.

0-255 arasında değerleri 50 ms aralıklarla sürekli bir artırarak D portundan DAC entegresine gönderiniz. Daha sonra 255'ten 0'a doğru değerleri bir azaltınız. DAC entegresinin çıkışına bağlı LED'in parlaklık değişimini gözlemleyiniz.

Kullanılacak Araç Gereç: Tablo 4.13'te belirtilmiştir.

Tablo 4.13: Uygulama 4.4 İçin Malzeme Listesi

Adı	Özellği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 adet
DAC entegresi	DAC0800, DIP soket	1 adet
Direnç	10 k Ω	3 adet
Direnç	1 k Ω	2 adet
Direnç	470 Ω	1 adet
Direnç	220 Ω	1 adet
Kondansatör	10 nF	1 adet
LED	Kırmızı	1 adet
NPN transistör	BC237	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı teli		1 m

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **DAC_Uyg_01** adlı yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```
#define F_CPU 8000000UL
```

```
#include <avr/io.h>
```

```
#include <util/delay.h>
```

```
int main(void)
```

```
{
```

```
    int forward = 1;
```

```
    DDRD = 0xFF;
```

```
    PORTD = 0x00;
```

```
    while (1)
```

```
    {
```

```
        if (forward == 1)
```

```
            PORTD++;
```

```
        else
```

```
            PORTD--;
```

```
        _delay_ms(75);
```

```
        if (PORTD == 255)
```

```
            forward = 0;
```

```
        if (PORTD == 0)
```

```
            forward = 1;
```

```
    }
```

```
}
```

```
// İleri doğru sayma değişkeni
```

```
// PORTD'yi çıkış olarak ayarla.
```

```
// PORTD'yi sıfırla.
```

```
// Sonsuz döngü
```

```
// İleri doğru sayma değişkeni 1 ise
```

```
// PORTD'nin değerini 1 artır.
```

```
// İleri doğru sayma değişkeni 0 ise
```

```
// PORTD'nin değerini 1 azalt.
```

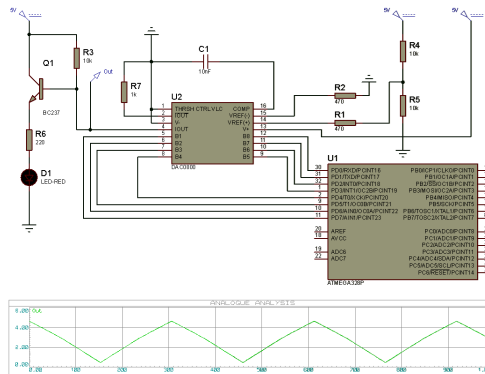
```
// 75 ms bekle.
```

```
// PORTD=0xFF ise geri doğru say.
```

```
// PORTD=0x00 ise ileri doğru say.
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğuna dikkat ediniz.

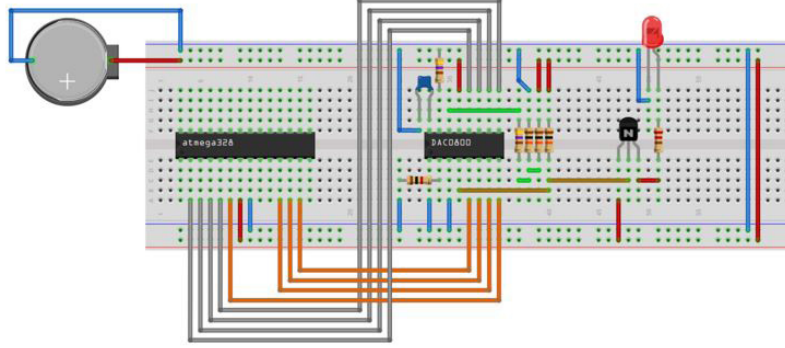
5. Adım: Devre simülasyon programında, Görsel 4.30'da verilen devreyi kurarak simüle ediniz. I_{OUT} çıkışının zamana göre değişimini izlemek için 0-1000 saniye aralığında analog analiz yapınız. Mikrodenetleyici yazılımı olarak bu uygulamada oluşturduğunuz HEX dosyasını tanımlayınız.



Görsel 4.30: Uygulama 4.4 için simülasyon devresi

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.31’de verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.31: Uygulama 4.4 için breadboard üzerine kurulan devre

8. Adım: DAC0800 entegresinin I_{OUT} pini çıkışındaki gerilimi, bir voltmetre yardımıyla ölçünüz. LED parlaklığının değişimini izleyiniz.

9. Adım: Güç kaynağını kapatınız.

10. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Simülasyon devresi çizilerek başarılı bir şekilde simülasyon gerçekleştirildi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. I_{OUT} çıkışındaki gerilim ve LED parlaklığının zamanla değişimi tespit edildi.		
6. Temizliğe dikkat edildi.		

4.1.5.2. Zamanlayıcı / Sayıcı Modülü ile PWM İşareti Üretme

Mikrodenetleyici ile DAC işlemi gerçekleştirmenin bir diğer yöntemi, **zamanlayıcı / sayıcı [Timer/Counter (Z/S)]** modülü kullanarak PWM işareti üretmektir. Bu sayede elektromekanik cihazlar ve analog devrelerin kontrolü için farklı gerilim çıkış düzeyleri oluşturulabilmektedir.

Z/S etkinleştirildiğinde, **TCNTn** değeri, Z/S'ye uygulanan saat darbesine göre birer birer artar veya azalır. Z/S'nin ulaştığı en büyük sayı değerine **TOP** değeri denir. Z/S, TOP değerine ulaştığında sıfırlanarak ayarlanan moda göre ileriye veya geriye doğru sayabilir.

TCNTn değeri, OCRnA veya OCRnB değerinden birine eşit olursa ilgili **Z/S kesmesi (OCnA veya OCnB)** etkinleştirilir. Bu sayede kesme anlarında zamanlanan işlemler yapılır. Ayrıca, dalga şekli üretimi için ilgili bitler programlanmışsa OCnA veya OCnB pinlerinden **darbe genişlik modülasyonu [pulse width modulation (PWM)]** yöntemiyle üretilen veya PWM'siz dalga şekli çıkışı alınır. **MAX** değeri, Z/S'nin sayabileceği en yüksek değerdir. MAX değeri, n bit sayısı olmak üzere $n = 8$ bitlik Z/S için

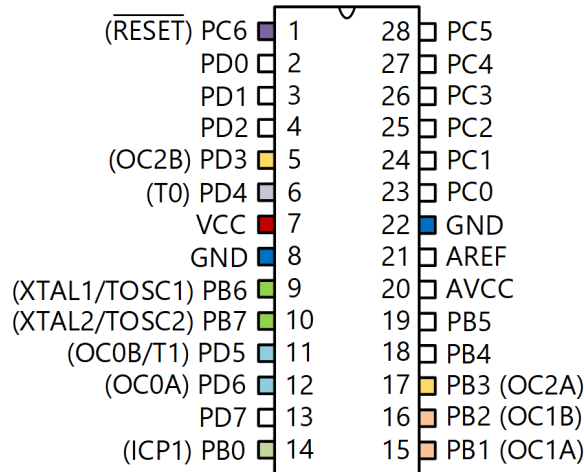
$$2^n - 1 = 2^8 - 1 = 255,$$

$n = 16$ bitlik Z/S için ise

$$2^n - 1 = 2^{16} - 1 = 65\,535 \text{ değerlerine eşittir.}$$

İlgili kontrol bitlerinin programlanması hâlinde, Z/S değeri, OCRnA veya OCRnB haricinde, Z/S'nin alabileceği en küçük değer (0, **BOTTOM**) veya en yüksek değer (MAX) ile de karşılaştırılabilir. Bu durumda, mantıksal kontrol devresi tarafından karşılaştırma gerçekleştirilir. Karşılaştırma sonucuna göre zamanlayıcı taşma kesmesi (**TOVn**) etkinleştirilir.

Atmega328P mikrodenetleyicinin Z/S modülleri ile ilgili DIP soket pin şeması Görsel 4.33'te verilmiştir. Burada, V_{cc} pininden +3,3 V ile +5 V arasında bir besleme gerilimi uygulanır. GND ise topraklamadır. RESET pini, mikrodenetleyiciye resetleme işlemi yapılacağı zaman lojik 0 yapılır. XTAL1/TOSC1 ve XTAL2/TOSC2 pinlerine haricî kristal osilatör veya diğer haricî osilatörler bağlanabilir. OCnx uçları, karşılaştırma eşleşmesine göre üretilen çıkış dalga şeklinin elde edildiği pinlerdir. İstenirse T0 ve T1 pinlerinden haricî tetikleme yapılarak ilgili Z/S değeri değiştirilebilmektedir. ICP1 ise giriş yakalama pinidir. Bu pinden uygulanan haricî kare dalga işaretin frekansı ve dalga şekli özellikleri mikrodenetleyici tarafından tespit edilebilmektedir.



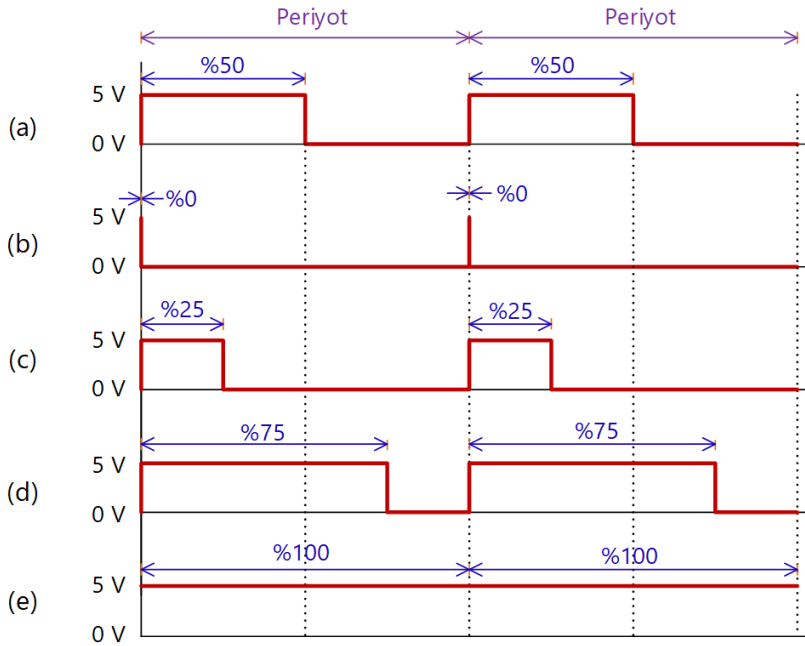
Görsel 4.33: Atmega328P mikrodenetleyicinin Z/S modülü ile ilişkili pinleri

4.1.7. Darbe Genişlik Modülasyonu (PWM)

Darbe genişlik modülasyonu [pulse width modulation (PWM)], kare dalga işaretin lojik 0 ve lojik 1 olduğu zaman dilimlerinin değiştirilmesidir. Bu teknik; işaret işleme, ışık seviyesinin ayarlanması, motor hız kontrolü gibi birçok uygulamada gerilim veya akım kontrolü sağlamak amacıyla kullanılmaktadır.

Kare dalga işaretin lojik 1 düzeyinde olduğu zamana, **darbe genişliği (pulse width)** adı verilir. PWM'nin amacı, darbe genişliğini değiştirmektir. Böylece darbe genişliği azaldığında, PWM işaretin ortalama gerilimi azalacak; arttığında ise ortalama gerilim yükselecektir. Anahtarlama ve yükselteç devrelerine uygulanan gerilim, PWM tekniği ile değiştirilerek bu devrelerin kontrolü sağlanmaktadır.

PWM tekniğinde, işaretin lojik 1 düzeyinde olduğu sürenin toplam süreye (periodya) oranına **görev döngüsü (duty cycle)** adı verilir. Örneğin, işaret periyodunun yarısı süresince lojik 1 düzeyinde olan bir kare dalga işaretin görev döngüsü değeri %50'dir. Görsel 4.34; (a)'da %50, (b)'de %0, (c)'de %25, (d)'de %75 ve (e)'de %100 görev döngüsü değerleri olan kare dalga işaretleri görülmektedir.



Görsel 4.34: Farklı görev döngüsü (duty cycle) değerleri için kare dalga işaretleri:
a %50, b %0, c %25, d %75, e %100 görev döngüsü

Mikrodenetleyicilerde PWM tekniği ile dalga şekillerinin üretilmesi için Z/S modülleri kullanılır. Genellikle işaretin periyodu, Z/S'nin MAX değeriyle görev döngüsü değeri ise zamanlayıcının TOP değeri ile belirlenir. Üretilen dalga şekli, OCnA veya OCnB pinlerinden alınır.

4.1.8. Z/S0 Modülü

Z/S0 modülü, 8 bitlik bir Z/S olup iki adet çıkış karşılaştırma birimine ve PWM desteğine sahiptir. Programın çalışması sırasında doğru zamanlama yapabilmek (olay yönetimi) ve dalga şekli üretmek amacıyla kullanılır. 0 ile 255 arasında sayma işlemi gerçekleştirir.

Z/S0 modülünde, Z/S (**TCNT0**) ve çıkış karşılaştırma (**OCR0A ve OCR0B**) adlı 8 bitlik kaydediciler bulunur. Zamanlayıcı kesme bayrak kaydedicisi (**TIFR0**) üzerinden, kesme istekleri takip edilebilir. Bu kesmeler, zamanlayıcı kesme maskeleme kaydedicisi (**TIMSK0**) üzerinden etkinleştirilebilir veya devre dışı bırakılabilir. Z/S kontrol kaydedicisi (**TCCR0A ve TCCR0B**) üzerinden sayıcı ayarlamaları yapılabilir.

Z/S0 modülü, dâhilî saat darbesiyle **ön ölçekleyici (prescaler)** ile ölçeklendirilen bir saat darbesiyle ya da **T0** pini üzerinden haricî olarak uygulanan saat darbesiyle çalışabilmektedir. Çıkış karşılaştırma kaydedicilerinin (OCR0A ve OCR0B) değerleri, zamanlayıcı değeriyle (TCNT0) sürekli karşılaştırılır. Eşleşme durumunda, ilgili karşılaştırma bayrağı (**OCF0A veya OCF0B**) 1 yapılır ve **OC0A** veya **OC0B** kesme isteği oluşturulur. Karşılaştırma sonucuna göre OC0A ve OC0B pinlerinden PWM dalga şekli veya değişik çıkış frekansına sahip işaretler elde edilebilir. Çıkış alabilmek için OC0A ve OC0B pinlerine karşılık gelen DDR (Data Direction Register) değerleri, çıkış (1) olarak ayarlanmalıdır.

4.1.9. Z/S0 Çalışma Modları

Çalışma modları, Z/S ve çıkış karşılaştırma pinlerinin çalışma şeklini belirler. Seçilen çalışma moduna göre her bir zamanlayıcı saat darbesinde, Z/S (TCNT0) artırılabilir, azaltılabilir veya sıfırlanabilir. Çalışma modunu tanımlamak için dalga şekli üretme modu (WGM0[2:0]) ve karşılaştırma çıkış modu (COM0A[1:0] ve COM0B[1:0]) bitleri, uygun şekilde ayarlanır.

4.1.9.1. Normal Mod

Z/S'nin en basit çalışma şekli, normal moddur. Normal mod için WGM0[2:0] bitleri 0 yapılır. Bu modda sayıcı, sürekli yukarıya doğru sayar ve sıfırlama işlemi gerçekleşmez. 8 bitlik sayı uzunluğunun en yüksek değerine ulaşıldığında (MAX = TOP = 0xFF = 255) sayıcı, tekrar başa döner ve 0x00 = 0 değerinden başlayarak yeniden saymaya devam eder. TCNT0, tekrar 0'dan başladığında **Z/S taşıma bayrağı (TOV0)** 1 olur. Z/S taşıma kesmesi çalıştığında TOV0 bayrağı sıfırlanır.

4.1.9.2. Karşılaştırma Eşleşmesinde Z/S'yi Sıfırlama (CTC) Modu

Karşılaştırma eşleşmesinde Z/S'yi sıfırlama (CTC) modunda, Z/S (TCNT0) değeri sürekli yukarıya doğru ilerler ve OCR0A değeri ile sürekli karşılaştırılır. TCNT0 ile OCR0A değeri birbirine eşitse sayıcı sıfırlanır ve tekrar baştan saymaya devam eder. Bu sırada OCF0A bayrağı 1 olur ve istenirse OCF0A kesmesi etkinleştirilebilir. OCR0A değeri, sayıcının üst değerini (TOP) belirler ve OCF0A kesme rutini içerisinde bu değer değiştirilebilir. CTC modunda, COM0A[1:0] = 01 olarak ayarlandığı takdirde, karşılaştırma çıkış bitleri, **toggle** yapılarak OC0 çıkış işaretinin frekansını değiştirmek mümkün olmaktadır. Üretilen dalga şekli frekansı aşağıdaki formül ile hesaplanabilir:

$$f_{OCnx} = \frac{f_{CLK_IO}}{2 \cdot N \cdot (1 + OCRnx)}$$

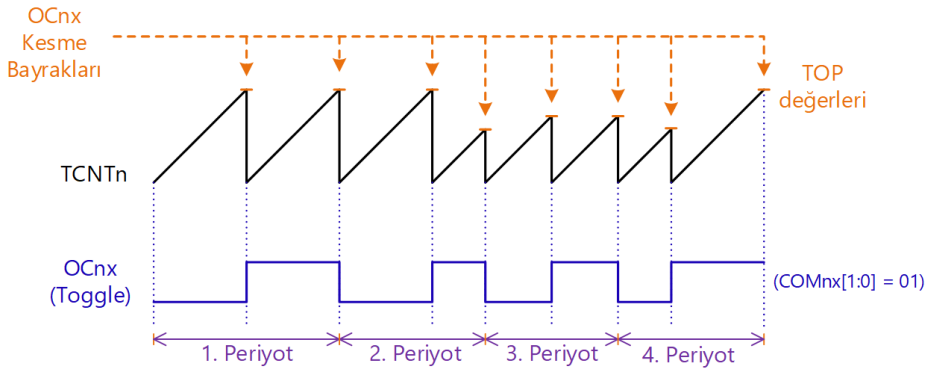
Burada f_{CLK_IO} , mikrodenetleyici çalışma saat frekansını, N , ön ölçekleme (prescaler) değerini (1, 8, 64, 256 veya 1 024 olabilir); $OCRnx$ ise OCR0A değerini ifade etmektedir.



NOT

Toggle durumu bir lojik değerin 0 ise 1, 1 ise 0 yapılmasıdır. Diğer bir ifadeyle lojik durumun terslenmesini ifade eder.

CTC modunda, değişken frekans değerine sahip çıkış dalga şekli üretilmesi Görsel 4.35'te verilen zamanlama diyagramında gösterilmiştir. Görsel 4.35'te her bir OC0A kesmesinde, OCR0A değeri farklı bir TOP değerine ayarlanmaktadır. Karşılaştırma çıkış bitleri için **toggle** modu etkin olduğunda farklı frekans değerlerine sahip dalga şekli, OC0A çıkışında elde edilir.



Görsel 4.35: CTC modu zamanlama diyagramı

4.1.9.4. Doğru Faz PWM Modu

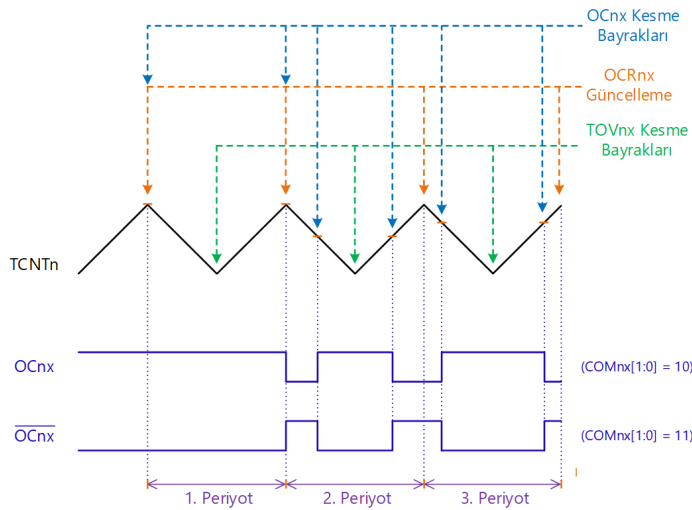
Doğru faz (phase correct) PWM modu, yüksek hassasiyetli PWM üretmek amacıyla kullanılır. Doğru faz PWM modunu etkinleştirmek için WGM[2:0] bitleri 001 veya 101 olarak ayarlanır. Z/S, sürekli olarak sıfırdan ileriye doğru TOP değerine kadar saydıktan sonra, TOP değerinden geriye sıfıra doğru sayar. TOP değeri, WGM[2:0] bitleri 001 iken MAX = 0xFF = 255; WGM[2:0] bitleri 101 iken OCR0A olarak ayarlanır.

Evirmeyen karşılaştırma çıkış modunda, Z/S ileriye doğru sayarken TCNT0 ile OCR0x eşit olduğunda çıkış karşılaştırma biti (OC0x) sıfırlanır. Sayıcı geriye doğru sayarken TCNT0 ile OCR0x eşit olduğunda ise 1 yapılır. Eviren karşılaştırma çıkış modunda ise bu çalışma şeklinin tersi geçerlidir. PWM dalga şekli üretimi, iki sayıcı kesme periyodunda gerçekleştiği için en yüksek çalışma frekansı hızlı PWM moduna göre daha düşüktür. Buna karşın, doğru faz PWM modunun simetrik dalga şekli üretmesi dolayısıyla daha çok motor hız kontrolü uygulamalarında tercih edilir.

Görsel 4.37’de doğru faz PWM modunun zamanlama diyagramı verilmiştir. TCNT0, OCR0x değerine eşit olduğu anlarda, OC0x kesme bayrağı 1 olur. OCR0x değerleri, sayıcı TOP değerine ulaştığında güncellenir. Z/S taşma kesmesi ise TCNT0 değeri sıfıra ulaştığında gerçekleşir. Üretilen dalga şeklinin OCnx pininde görülebilmesi için ilgili port bitinin DDR değeri çıkış olarak ayarlanmalıdır. Doğru faz PWM için çıkış frekansı aşağıdaki formül kullanılarak hesaplanabilir:

$$f_{OCnx_PCPWM} = \frac{f_{CLK_IO}}{n \cdot 510}$$

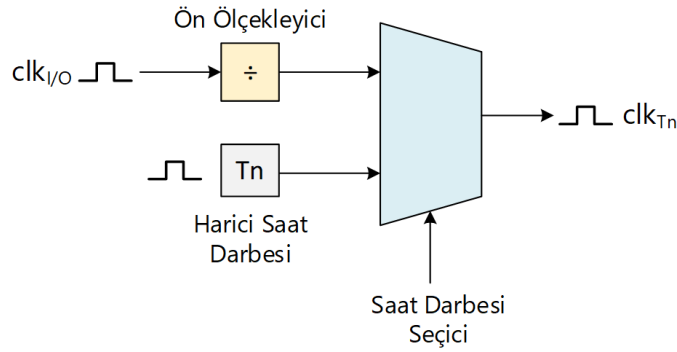
Burada f_{CLK_IO} , mikrodenetleyici çalışma saat frekansını, N ise ön ölçekleme (prescaler) değerini (1, 8, 64, 256 veya 1 024 olabilir) ifade etmektedir.



Görsel 4.37: Doğru faz PWM modu zamanlama diyagramı

4.1.10. Ön Ölçekleyici (Prescaler) Kullanımı

Ön ölçekleyici (prescaler), mikrodenetleyici çalışma frekansının bölünmesi ve Z/S'yi daha düşük frekanslarda çalıştırmak amacıyla kullanılır. Bu sayede Z/S'nin sayma hızı değiştirilebilmektedir. Görsel 4.38'de Z/S modülü için saat darbesi kaynağının seçimini anlatan şema verilmiştir. Burada, $clk_{I/O}$, mikrodenetleyici çalışma saat darbesidir. T_n ise mikrodenetleyici haricî saat darbesi pinidir. n indisi Z/S modül numarasını ifade etmek üzere; Z/S0 modülü için pin adı **T0**, Z/S1 modülü için pin adı **T1** olur. clk_{Tn} , Z/S modülünün çalışma saat darbesini ifade eder. Saat darbesi kaynağı, Z/S saat seçim bitleri (CSn[2:0]) ile seçilebilmektedir.



Görsel 4.38: Z/S saat darbesi kaynağını seçme

Mikrodenetleyici çalışma frekansı ($f_{I/O}$), ön ölçekleyici yardımıyla 1, 8, 64, 256 veya 1024'e bölünerek Z/S modülünün çalışma frekansı (f_{CLK_Tn}) belirlenebilir. Bu işleme **ön ölçekleme (prescaling)** adı verilir.

Örneğin; mikrodenetleyicinin 1 MHz'lik bir frekans ile çalıştığını varsaydığımızda ön ölçekleyici değeri 8 olduğunda, Z/S çalışma frekansı,

$$f_{CLK_Tn} = f_{I/O} / 8 = 1\,000\,000\text{ Hz} / 8 = 125\,000\text{ Hz} = 125\text{ kHz olarak hesaplanır.}$$

Z/S çalışma periyodu (T_{CLK_Tn}) ise

$$T_{CLK_Tn} = 1 / f_{CLK_Tn} = 1 / 125000 = 8\text{ }\mu\text{s olarak bulunur.}$$

Bu durumda, Z/S değerinin 1 artması için gereken süre 8 μs 'dir.



SIRA SİZDE

4 MHz çalışma frekansına sahip bir mikrodenetleyicide, Z/S çalışma periyodunun 256 μs olabilmesi için ön ölçekleyici değeri kaç olmalıdır? Hesaplayınız.

4.1.11. Z/S0 Modülünde Karşılaştırma Çıkış Modu ve Çalışma Modunun Belirlenmesi

Z/S0 modülü için karşılaştırma çıkış modu ve çalışma modunun belirlenmesi işlemi, Z/S kontrol kaydedici A (TCCR0A) üzerinden gerçekleştirilir. TCCR0A kaydedicisinin bitleri, Görsel 4.39'da verilmiştir.

	7	6	5	4	3	2	1	0
TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

Görsel 4.39: TCCR0A bitleri

7.-6. Bitler COM0A[1:0]: Karşılaştırma eşleşmesi çıkışı, A modu bitleridir. Bu bitler, OC0A pininin davranışını kontrol eder. OC0A pini, COM0A[1:0]=00 olması hâlinde normal port pini olarak çalışır. Diğer durumlarda ise Z/S'nin çalışma moduna göre farklı işlevler yüklenir.

OC0A, normal veya CTC çalışma modlarında Tablo 4.14'te verilen işlevlere sahiptir.

Tablo 4.14: PWM Olmayan Durumlar İçin Çıkış Karşılaştırma Modu

COM0A1	COM0A0	Tanım
0	0	Normal port çalışması OC0A bağlantısı devre dışı.
0	1	Karşılaştırma eşleşmesi hâlinde OC0A için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesi hâlinde OC0A'yı sıfırla.
1	1	Karşılaştırma eşleşmesi hâlinde OC0A'yı 1 yap.



NOT

Karşılaştırma eşleşmesi (compare match), TCNT0 ile OCRA0'ın karşılaştırılması sonucunda eşit olması durumudur.

OC0A, hızlı PWM modunda Tablo 4.15'te verilen işlevlere sahiptir.

Tablo 4.15: Hızlı PWM Modu İçin Çıkış Karşılaştırma Modu

COM0A1	COM0A0	Tanım
0	0	Normal port çalışması OC0A bağlantısı devre dışı.
0	1	WGM02=0 iken normal port çalışması OC0A bağlantısı devre dışı. WGM02=1 iken karşılaştırma eşleşmesinde OC0A için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesinde OC0A'yı sıfırla, sayıcı BOTTOM (0) değerine ulaştığında OC0A'yı 1 yap (evirmeyen mod).
1	1	Karşılaştırma eşleşmesinde OC0A'yı 1 yap, sayıcı BOTTOM (0) değerine ulaştığında OC0A'yı sıfırla (eviren mod).

OC0A, doğru faz PWM modunda Tablo 4.16'da verilen işlevlere sahiptir.

Tablo 4.16: Doğru faz PWM Modu İçin Çıkış Karşılaştırma Modu

COM0A1	COM0A0	Tanım
0	0	Normal port çalışması OC0A bağlantısı devre dışı.
0	1	WGM02=0 iken normal port çalışması OC0A bağlantısı devre dışı. WGM02=1 iken karşılaştırma eşleşmesinde OC0A için toggle işlemi yap.
1	0	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC0A'yı sıfırla. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC0A'yı 1 yap.
1	1	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC0A'yı 1 yap. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC0A'yı sıfırla.

5.-4. Bitler COM0B[1:0]: Karşılaştırma eşleşmesi çıkışı, B modu bitleridir. Bu bitler, OC0B pininin davranışını kontrol eder. OC0B pini, COM0B[1:0] = 00 olması hâlinde, normal port pini olarak çalışır. Diğer durumlarda ise Z/S'nin çalışma moduna göre farklı işlevler yüklenir.

OC0B, normal veya CTC çalışma modlarında Tablo 4.17'de verilen işlevlere sahiptir.

Tablo 4.17: PWM Olmayan Durumlar İçin Çıkış Karşılaştırma Modu

COM0B1	COM0B0	Tanım
0	0	Normal port çalışması OC0B bağlantısı devre dışı.
0	1	Karşılaştırma eşleşmesi hâlinde OC0B için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesi hâlinde OC0B'yi sıfırla.
1	1	Karşılaştırma eşleşmesi hâlinde OC0B'yi 1 yap.

OC0B, hızlı PWM modunda Tablo 4.18'de verilen işlevlere sahiptir.

Tablo 4.18: Hızlı PWM Modu İçin Çıkış Karşılaştırma Modu

COM0B1	COM0B0	Tanım
0	0	Normal port çalışması OC0B bağlantısı devre dışı.
0	1	<i>Rezerve</i>
1	0	Karşılaştırma eşleşmesinde OC0B'yi sıfırla, sayıcı BOTTOM (0) değerine ulaştığında OC0B'yi 1 yap (evirmeyen mod).
1	1	Karşılaştırma eşleşmesinde OC0B'yi 1 yap, sayıcı BOTTOM (0) değerine ulaştığında OC0B'yi sıfırla (eviren mod).

OC0B, doğru faz PWM modunda Tablo 4.19’da verilen işlevlere sahiptir.

Tablo 4.19: Doğru Faz PWM Modu İçin Çıkış Karşılaştırma Modu

COM0B1	COM0B0	Tanım
0	0	Normal port çalışması OC0B bağlantısı devre dışı.
0	1	<i>Rezerve</i>
1	0	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC0B’yi sıfırla. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC0B’yi 1 yap.
1	1	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC0B’yi 1 yap. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC0B’yi sıfırla.

1-0. Bitler WGM01 ve WGM00: Bu bitler, Z/S’nin çalışma modunu ayarlamak için kullanılır. TCCR0B kaydedicisinde bulunan WGM02 bitiyle birlikte işlev yaparlar. Bu bitler; sayıcının sayma sürecini kontrol eder, en yüksek (TOP) sayıcı değerini belirler ve nasıl bir dalga şekli üretileceğini ayarlar. WGM02, WGM01 ve WGM00 bitlerinin işlevi Tablo 4.20’de verilmiştir (MAX = 0xFF, BOTTOM = 0x00).

Tablo 4.20: Z/S0 Çalışma Modları

Mod	WGM02	WGM01	WGM00	Çalışma Modunun Adı	TOP	OCR0x Güncelleme	TOV Bayrağı
0	0	0	0	Normal	0xFF	Derhâl	MAX
1	0	0	1	Doğru faz PWM	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Derhâl	MAX
3	0	1	1	Hızlı PWM	0xFF	BOTTOM	MAX
4	1	0	0	<i>Rezerve</i>	-	-	-
5	1	0	1	Doğru faz PWM	OCRA	TOP	BOTTOM
6	1	1	0	<i>Rezerve</i>	-	-	-
7	1	1	1	Hızlı PWM	OCRA	BOTTOM	TOP

4.1.12. Z/S0 Modülünde Zorlamalı Çıkış Karşılaştırma ve Çalışma Saat Darbesi Seçimi

Normal veya CTC çalışma modlarında, OC0A veya OC0B’de üretilen dalga şeklinin zorlamalı olarak değiştirilmesi ve Z/S çalışma saat darbesi seçimi için Z/S kontrol kaydedicisi B (TCCR0B) kullanılır. TCCR0B bitleri Görsel 4.40’ta verilmiştir.

	7	6	5	4	3	2	1	0
TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

Görsel 4.40: TCCR0B bitleri

7-6. Bitler FOC0A ve FOC0B: Bu bitler 1 yapıldığında ilgili dalga şekli üretim biriminde, otomatik olarak karşılaştırma eşleşmesi durumu oluşturulur. COM0x[1:0] bitlerinin durumuna göre OC0A veya OC0B pini çıkışı değiştirilir ancak bu bitlerin değiştirilmesi OC0x kesmesini tetiklemez.

2-0. Bitler CS0[2:0]: Bu bitler, çalışma saat seçimi bitleridir. Z/S için saat darbesi kaynağını ve ön ölçekleyici değerini belirler. CS0 bitlerinin anlamları, Tablo 4.21'de verilmiştir.

Tablo 4.21: Saat Darbesi Kaynağı Seçimi

CS02	CS01	CS00	Tanım
0	0	0	Saat darbesi kaynağı yok (Zamanlayıcı / sayıcı çalışmaz)
0	0	1	clk_{IO} (Ön ölçekleme yok)
0	1	0	$clk_{IO}/8$ (Ön ölçekleyici)
0	1	1	$clk_{IO}/64$ (Ön ölçekleyici)
1	0	0	$clk_{IO}/256$ (Ön ölçekleyici)
1	0	1	$clk_{IO}/1024$ (Ön ölçekleyici)
1	1	0	Harici saat kaynağı pini (T0) (düşen kenar tetiklemeli)
1	1	1	Harici saat kaynağı pini (T0) (yükselen kenar tetiklemeli)

4.1.13. Z/S0 Modülü Z/S Değeri

Z/S değeri, 8 bitlik TCNT0 kaydedicisi içerisinde yer alır. Çalışma anında sayıcı, her bir Z/S saat darbesinde TCNT0'ın değerini 1 artırır veya azaltır. TCNT0'ın değeri, karşılaştırma eşleşmesi anında değiştirilemez. Diğer durumlarda değiştirilebilir ancak TCNT0'ın Z/S modülü çalışırken değiştirilmesi karşılaştırma eşleşmesinin kaçırılmasına neden olabileceğinden önerilmez.

4.1.14. Z/S0 Modülü Çıkış Karşılaştırma Kaydedicileri

Çıkış karşılaştırma (output compare) kaydedicileri (OCR0A ve OCR0B), TCNT0 sayıcı değeri ile karşılaştırılacak değeri tutan 8 bitlik kaydedicilerdir. Karşılaştırma eşleşmesi durumunda, çıkış karşılaştırma kesmesi oluşur ya da OC0A veya OC0B pinlerinde üretilen dalga şekli değişir.

4.1.15. Z/S0 Modülünde Z/S Kesmelerinin Etkinleştirilmesi

Z/S kesmelerinin etkin yapılması için Z/S kesme maskeleme kaydedicisi (TIMSK0) bitleri ayarlanır. TIMSK0 bitleri, Görsel 4.41'de verilmiştir.

	7	6	5	4	3	2	1	0
TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0

Görsel 4.41: TIMSK0 bitleri

2. Bit OCIE0B: OCIE0B biti 1 yapıldığında Z/S karşılaştırma eşleşmesi B (OC0B) kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

1. Bit OCIE0A: OCIE0A biti 1 yapıldığında Z/S karşılaştırma eşleşmesi A (OC0A) kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

0. Bit TOIE0: TOIE0 biti 1 yapıldığında Z/S taşma kesmesi (TOV0) etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

4.1.16. Z/S0 Modülünde Z/S Kesme Bayraklarının Kontrolü

Z/S kesmelerinden biri oluştuğunda ilgili kesme bayrağı biti 1 olur. Bu bayrak biti, kesme vektörüne gidildiğinde donanım tarafından otomatik olarak sıfırlanır. Yazılım içerisinde kesme bayrak bitlerini sıfırlamak için bu bitlere 1 yazılmalıdır. Z/S kesme bayrak bitleri TIFR0 kaydedicisinden kontrol edilir. Görsel 4.42’de TIFR0 bitleri verilmiştir.

	7	6	5	4	3	2	1	0
TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0

Görsel 4.42: TIFR0 bitleri

2. Bit OCF0B: OCR0B ile TCCR0’ın değerleri eşit olduğunda (karşılaştırma eşleşmesi durumunda) karşılaştırma eşleşmesi bayrağı (OCF0B) 1 yapılır.

1. Bit OCF0A: OCR0A ile TCCR0’ın değerleri eşit olduğunda (karşılaştırma eşleşmesi durumunda) karşılaştırma eşleşmesi bayrağı (OCF0A) 1 yapılır.

0. Bit TOV0: TCCR0 değeri en yüksek değere ulaştıktan sonra, taşma bayrağı TOV0 1 yapılır. Kesmeler etkinse zamanlayıcı taşma kesmesi vektörüne gidilir.

4.1.17. Z/S0 Modülü Çalıştırma Adımları

Z/S0 modülü, seçilen çalışma moduna göre ayarlanır. Kodlama adımları aşağıda sıralanmıştır:

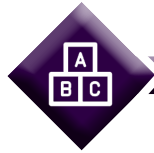
1. Öncelikle Z/S çalışma modu belirlenir. Bunun için TCCR0A ve TCCR0B kaydedicilerinde yer alan WGM0[2:0] bitleri Tablo 4.20’ye göre ayarlanır.

2. Çıkış karşılaştırma pininin işlevi belirlenir. Bunun için TCCR0A kaydedicisinde yer alan COM0A[1:0] bitleri ayarlanır. Bu bitler çalışma moduna göre farklı işlevlere sahiptir. Normal ve CTC çalışma modlarında Tablo 4.14 ve 4.17’ye göre ayarlama yapılır. Hızlı PWM çalışma modunda, Tablo 4.15 ve 4.18’e göre bitler belirlenir. Doğru faz PWM için ise Tablo 4.16 ve 4.19’a göre işlem gerçekleştirilir.

3. TCNT0 kaydedicisi sıfırlanır. OCR0x kaydedicisine karşılaştırma değeri atanır.
4. Kullanılacak kesme kaynakları belirlenir. Bunun için öncelikle **sei()** komutu ile SREG kaydedicisinde I biti 1 yapılır ve kesmeler etkinleştirilir. Daha sonra TIMSK0 kaydedicisinde yer alan bitler kullanılarak ilgili kesmeler etkin yapılır.
5. Saat darbesi kaynağı ve ön ölçekleme değeri belirlenir. Bunun için TCCR0B kaydedicisinde yer alan CS0[2:0] bitleri Tablo 4.21'e göre ayarlanır. Bu işlemle birlikte, Z/S çalışmaya başlar.

**NOT**

Z/S çalışırken istenirse main fonksiyonunda yer alan *while (1) { }* bloku içerisinde başka işlemler de yapılabilir. Bu işlemler, sayma işlemini etkilemez. Z/S kullanmanın en büyük avantajlarından biri bu özelliğidir.

**UYGULAMA**

Adı:	LED'li Zamanlama Devresi	No : 4.5
AMAÇ:	Z/S Modülü kullanarak LED'li zamanlama devresini tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda Z/S0 modülü ile PC0 pinine bağlı kırmızı bir LED'in 500 ms süreyle yanıp sönmelerini sağlayan bir zamanlama devresi tasarlayınız. Z/S çıkış karşılaştırma modu sonucunu OC0A pinine bağlı, sarı bir LED üzerinden izleyiniz.

Kullanılacak Araç Gereç: Tablo 4.22'de belirtilmiştir.

Tablo 4.22: Uygulama 4.5 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 adet
Direnç	180 Ω	2 adet
LED	5mm, kırmızı ve sarı	2 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **Timer_Uyg_01** adlı yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: Z/S için çalışma saat darbesi frekansını ve ön ölçekleyici değerini hesaplayınız.

Mikrodenetleyici çalışma frekansını, $f_{I/O}=1$ MHz olarak belirleyiniz. Bunun için programlama sırasında **8 MHz dâhilî RC osilatör** ve **CKDIV8** sigortalarını etkin yapınız.

Ön ölçekleyici değerini 1024 olarak belirlediğimizde zamanlayıcı çalışma frekansı,

$$f_{TMR_CLK} = f_{I/O} / 1024 = 1\,000\,000\text{ Hz} / 1024 = 976,56\text{ Hz}$$

olarak hesaplanır. Bu durumda sayıcının değerinin 1 artması için geçen süre (periyot)

$$T_{TMR_CLK} = 1 / f_{TMR_CLK} = 1 / 976,56\text{ Hz} = 0,001024\text{ s} = 1,024\text{ ms}$$

olarak bulunur. Sayıcının 250 ms süresince sayması için en yüksek sayıcı değeri,

$$TOP = 250\text{ ms} / 1,024\text{ ms} = 244,14 \approx 244$$

bulunur. Sayıcının **250 ms × 2 = 500 ms** süresince sayması için sayıcının 2 kez 244 değerine kadar sayması gereklidir. Bunun için Z/S taşma kesmesi rutininde, sayma işleminin 2 kez olup olmadığı kontrol edilir. Sonuç olarak Z/S 1 MHz mikrodenetleyici çalışma frekansında, 1 024 ön ölçekleme değeri ile 2 kez 244'e kadar saymalıdır. Bu durumda 500 ms'ye yaklaşık bir değer elde edilir:

$$[1,024\text{ ms}] \times 244 \times 2 = 499,712\text{ ms}$$

4. Adım: Z/S çalışma modunu belirleyiniz. Burada TOP değerini istenen değere ayarlamak için CTC çalışma modunu tercih edebilirsiniz.

5. Adım: **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define LED PORTC                                // LED'in bağlı olduğu pin
int sayac = 0;                                   // Zamanlayıcı kesme sayısı değişkeni
ISR(TIMERO_COMPA_vect)                          // Timer/Counter0 karşılaştırma kesmesi
{
    cli();                                       // Global kesmeler devre dışı.
    if (sayac == 1)                             // sayac=1 ise
    {
        PORTC ^= (0x01 << LED); // LED değerini tersle.
        sayac = 0;                    // sayac değişkenini sıfırla.
    }
}
```

```

else // Aksi durumda,
{
    sayac++; // sayac=0 ise 1 artır.
}
sei(); // Global kesmeler etkin.
}

int main(void)
{
    DDRC = 0b00000001; // PORTC0 çıkış portu (LED için)
    DDRD = 0b01000000; // PORTD6 çıkış portu (OC0A pini için)

    TCCR0B &= ~(1 << WGM02); // Çalışma modu seçimi:
    TCCR0A |= (1 << WGM01); // WGM0[2:0]=010
    TCCR0A &= ~(1 << WGM00); // CTC çalışma modu
    TCNT0 = 0; // Z/S'yi sıfırla.
    OCR0A = 244; // Çıkış karşılaştırma kaydedici A

    TCCR0A &= ~(1 << COM0A1); // OC0A karşılaştırma eşleşmesinde toggle:
    TCCR0A |= (1 << COM0A0); // COM0A[1:0]=01

    TCCR0A &= ~(1 << COM0B1); // OC0B devre dışı: normal port
    TCCR0A &= ~(1 << COM0B0); // COM0B[1:0]=00

    sei(); // Global kesmeler etkin. SREG I=1
    TIMSK0 |= (1 << OCIE0A); // Çıkış karşılaştırma kesmesi A etkin.

    TCCR0B |= (1 << CS02) | (1 << CS00); // CS0[2:0]=101 (Z/S etkin.)

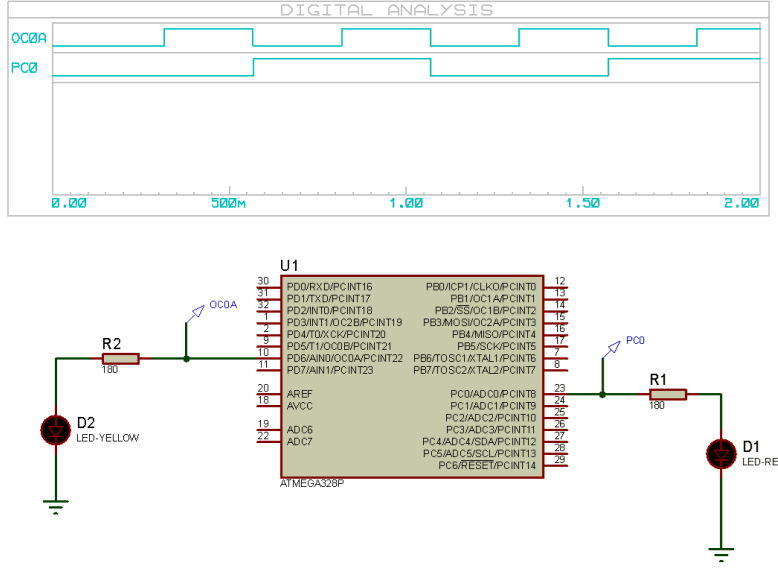
    TCCR0B &= ~(1 << CS01); // clk_I/O / 1024 (Ön ölçekleyici: /1024)

    while (1); // Sürekli çalış (sonsuz döngü).
}

```

6. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunacağını hatırlayınız.

7. Adım: Devre simülasyon programında, Görsel 4.43'te verilen devreyi kurarak simüle ediniz.

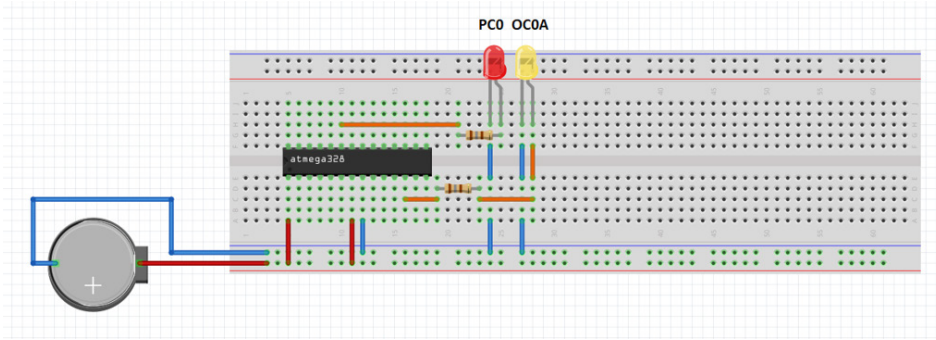


Görsel 4.43: Uygulama 4.5 için simülasyon devre şeması

8. Adım: PD6 pininde bulunan OC0A çıkışının ve PC0 pinine bağlı LED çıkışının dalga şeklini, simülasyon programının dijital analiz özelliği ile analiz ediniz. OC0A çıkışı her bir karşılaştırma eşleşmesi kesmesinde **toggle** olmaktadır. İki karşılaştırma eşleşmesi kesmesi süresince, LED yanmakta aynı süre kadar LED sönmektedir.

9. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

10. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.44'te gösterildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.44: Uygulama 4.5 için breadboard üzerindeki devre

11. Adım: OC0A ve PORTC0 pinlerinden üretilen dalga şeklini bir osiloskop cihazı ile izleyiniz. Osiloskopu kullanırken öğretmeninizden yardım alınız.

12. Adım: Güç kaynağını kapatınız.

13. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

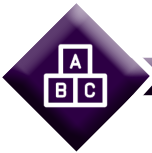
KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Devre simülasyonu üzerinde PC0 ve OC0A çıkışlarındaki işaretler izlendi.		
3. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Kırmızı LED'in 500 ms zaman aralığında yanıp sönmesi bir kronometre yardımıyla kontrol edildi.		
6. Her bir karşılaştırma eşleşmesinde sarı LED'in 250 ms zaman aralığında yanıp sönmesi bir kronometre yardımıyla kontrol edildi.		
7. Temizliğe dikkat edildi.		



SIRA SİZDE

4.5 No.lu uygulamada yaptığınız LED'li zamanlama devresinde PORTC'nin 0. bitine bağlı kırmızı LED'in yanıp sönme süresini, yaklaşık 1 saniye olacak şekilde ayarlayan mikrodenetleyici programını yazınız ve simülasyon programında simüle ediniz. Devresini kurarak çalıştırınız.



UYGULAMA

Adı:	PWM Dalga Şekli Üretme	No : 4.6
AMAÇ:	Z/S Modülü kullanarak PWM dalga şekli üreten devreyi tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda Z/S0 modülü ile yaklaşık 60 Hz frekansında %25 ve %75 görev döngülerine (duty cycle) sahip PWM dalga şekillerini OC0A ve OC0B pinlerinde eşzamanlı olarak üretiniz.

Kullanılacak Araç Gereç: Tablo 4.23'te belirtilmiştir.

Tablo 4.23: Uygulama 4.6 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **Timer_Uyg_02** adlı yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: Z/S için ön ölçekleyici değerini ve %25 ile %75 görev döngüleri için OCR0A ve OCR0B değerlerini hesaplayınız.

Mikrodenetleyici çalışma frekansını, $f_{i/o} = 1 \text{ MHz}$ olarak belirleyiniz. Bunun için programlama sırasında **8 MHz dâhilî RC osilatör** ve **CKDIV8** sigortalarını etkin yapınız.

Hızlı PWM modunda ön ölçekleyici değerini $N = 64$ olarak belirlediğimizde zamanlayıcı çalışma frekansı,

$$f_{OC0APWM} = \frac{f_{CLK_{IO}}}{N \cdot 256} = \frac{1000000}{64 \cdot 256} \cong 61.03 \text{ Hz}$$

Sonuç olarak Z/S, 1 MHz mikrodenetleyici çalışma frekansında, $N = 64$ ön ölçekleme değeri ile saymalıdır. Bu durumda OC0A ve OC0B çıkışlarında yaklaşık 60 Hz frekansına sahip dalga şekli elde edilir.

Sayıcı 0-255 arasında değer alacağından, %25 görev döngüsü için $256 \times 0,25=64$ olduğundan ve saymaya 0 ile başlanacağından OCR0A=63 değerini kullanabiliriz.

%75 görev döngüsü için $256 \times 0,75=192$ olduğundan ve saymaya 0 ile başlanacağından OCR0B=191 değerini kullanabiliriz.

4. Adım: Z/S çalışma modunu belirleyiniz. Burada, hızlı PWM çalışma modunu tercih edebilirsiniz.

5. Adım: main.c dosyası içerisinde aşağıda verilen program kodlarını yazınız.

```
#include <avr/io.h>

int main(void)
{
    DDRD = 0b01100000;           // PORTD5 ve PORTD6 çıkış (OC0B ve OC0A için)
    TCCR0B &= ~(1 << WGM02);     // Çalışma modu seçimi:
    TCCR0A |= (1 << WGM01) | (1 << WGM00); // WGM0[2:0]=011 Hızlı PWM
                                     // Z/S'yi sıfırla.
    TCNT0 = 0;                   // Çıkış karşılaştırma kaydedici A (%25 duty)
    OCR0A = 63;                  // Çıkış karşılaştırma kaydedici B (%75 duty)
    OCR0B = 191;

    TCCR0A |= (1 << COM0A1);     // OC0A - Hızlı PWM Evirmeyen Mod
    TCCR0A &= ~(1 << COM0A0);   // COM0A[1:0]=10

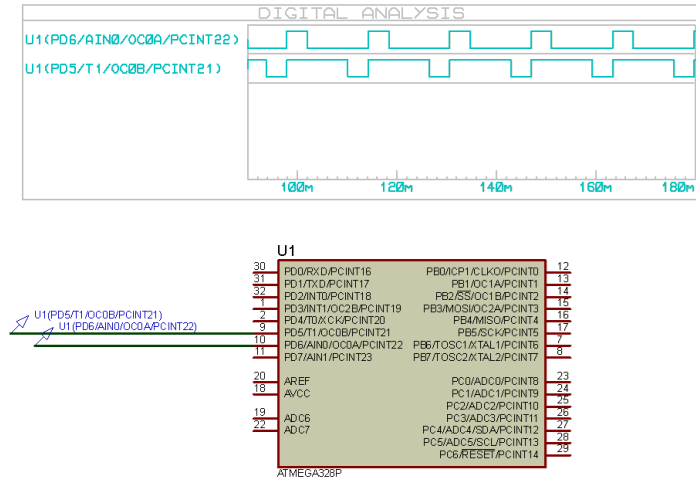
    TCCR0A |= (1 << COM0B1);     // OC0B - Hızlı PWM Evirmeyen Mod
    TCCR0A &= ~(1 << COM0B0);   // COM0B[1:0]=10

    TCCR0B &= ~(1 << CS02);     // CS0[2:0]=011 (Z/S etkin.)
    TCCR0B |= (1 << CS01) | (1 << CS00); // clk_I/O / 64 (Ön ölçekleyici: /64)

    while (1);                  // Sürekli çalış (sonsuz döngü).
}
```

6. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğunu hatırlayınız.

7. Adım: Devre simülasyon programında, Görsel 4.45'te verilen devreyi kurarak simüle ediniz.

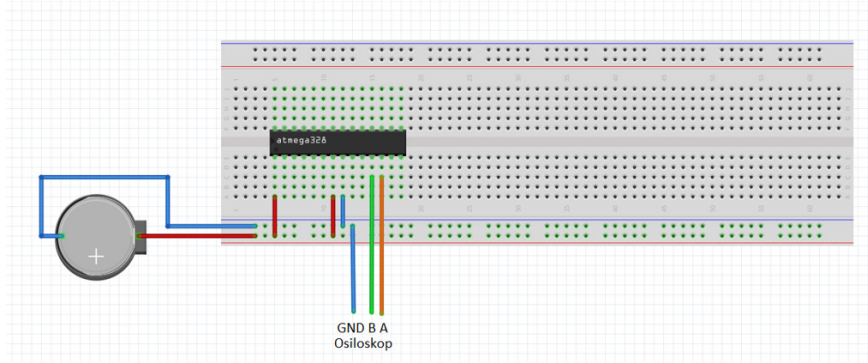


Görsel 4.45: Uygulama 4.6 için simülasyon devre şeması

8. Adım: PORTD6 pininde bulunan OC0A çıkışının ve PD5 pininde bulunan OC0B çıkışının dalga şeklini, simülasyon programının dijital analiz özelliği ile analiz ediniz. OC0A çıkışında %25 görev döngüsüne sahip, OC0B çıkışında ise %75 görev döngüsüne sahip PWM dalga şekilleri üretilmektedir.

9. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

10. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.46'te gösterildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.46: Uygulama 4.6 için breadboard üzerindeki devre

11. Adım: OC0A ve OC0B pinlerinden üretilen dalga şeklini bir osiloskop cihazı ile izleyiniz. Osiloskopu kullanırken öğretmeninizden yardım alınız.

12. Adım: Güç kaynağını kapatınız.

13. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Devre simülasyonu üzerinde OC0A ve OC0B çıkışlarındaki işaretler izlendi.		
3. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Osiloskop üzerinde OC0A ve OC0B çıkışlarındaki işaretler doğru bir şekilde görüntülendi.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

4.6 No.lu uygulamada verilen devreyi kullanarak %15 ve %85 görev döngüsü değerleri olan yaklaşık 60 Hz'lik PWM dalga şekillerini, OCOA ve OCOB pinlerinde üreten gömülü yazılımı kodlayınız.

4.1.18. Z/S1 Modülü

Z/S1 modülü, 16 bitlik çözünürlüğe sahip bir Z/S olup iki adet çıkış karşılaştırma birimi ve bir adet giriş yakalama (input capture) birimi bulunmaktadır. 0 ile 65 535 arasında sayma işlemi gerçekleştirir. Çalışma şekli, Z/S0 modülüne benzemekte olup kaydedici isimleri ve Z/S özelliklerinde farklılıklar bulunur.

Z/S kaydedicisi (TCNT1), çıkış karşılaştırma kaydedicileri (OCR1A/B) ve giriş yakalama (capture) kaydedicisi (ICR1), 16 bit uzunluğundadır. Z/S kontrol kaydedicileri (TCCR1A/B) ise 8 bit uzunluğundadır. Kesme istekleri, Z/S kesme bayrak kaydedicisinden (TIFR1) görülebilir. Kesme kontrolü ise Z/S kesme maskeleyici kaydedicisinde (TIMSK1) yer alır.

Z/S1 modülü, dâhilî osilatör ile ön ölçekleyici üzerinden veya haricî osilatör pininden (T1) tetiklenerek çalıştırılabilir. Tetikleme yapılmadığı takdirde, Z/S çalışmaz. Sayıcının ileriye, geriye doğru sayması sırasında OCR1A'nın içeriği ile TCCR1'in içeriği sürekli kontrol edilir ve OC1A/B çıkış pinlerinde karşılaştırma sonucuna göre PWM veya değişken frekanslı dalga şekli üretilebilir.

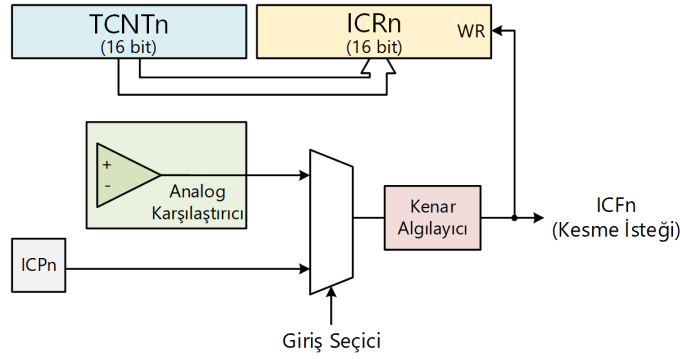
Z/S1 modülü, bir giriş yakalama birimi de içerir. Bu birimde yer alan giriş yakalama kaydedicisi (ICR1), giriş yakalama pini (ICP1) veya analog karşılaştırıcı modülü üzerinden uygulanan kare dalga işaretin, tetikleme anlarında yakalanan Z/S değerlerini tutar. Giriş yakalama birimi aynı zamanda, işaretin gürültüsünü engellemek için gürültü engelleyiciye sahiptir.

Z/S'nin TOP değeri, bazı çalışma modlarında OCR1A, ICR1 veya sabit bir değer olarak tanımlanabilir. Doğrudan kaydedici değişken adlarını yazarak 16 bitlik kaydedicilere okuma / yazma işlemi yapılabilir.

4.1.19. Z/S1 Giriş Yakalama Birimi

Z/S1 giriş yakalama birimi, ICP1 pininden veya analog karşılaştırıcı biriminden uygulanan işaretin frekansını, görev döngüsünü (duty cycle) ve diğer özelliklerini belirlemek amacıyla kullanılan birimdir. Giriş yakalama biriminin basitleştirilmiş blok şeması, Görsel 4.47'de verilmiştir.

Giriş seçici değerine göre ICP1 pininden ya da analog karşılaştırıcı biriminden uygulanan giriş işaretinin lojik düzeyi değiştiğinde (bir olay olduğunda) bu değişiklik, **kenar algılayıcı birimi** tarafından algılanır ve yakalama işlemi tetiklenir. Yakalama işlemi tetiklendiğinde giriş yakalama kaydedicisinin (ICR1) yazma girişi (WR) yetkilendirilir ve bunun sonucunda 16 bitlik TCNT1 değeri, 16 bitlik ICR1 kaydedicisine yazılır. Bu işlem sırasında, giriş yakalama bayrağı (ICF1) 1 yapılır. Giriş yakalama kesmesi (ICIE) biti 1 olarak ayarlandıysa ilgili kesme vektörüne gidilir ve kesme çalıştırılır. Kesme işlemi çalıştığında ICF1 bayrağı otomatik olarak sıfırlanır. Ayrıca ICF1 bayrağı yazılım ile 1 yapılarak da sıfırlanabilir.



GörSEL 4.47: Giriş yakalama birimi basitleştirilmiş blok şeması

Giriş yakalama birimi, Z/S normal çalışma modunda kullanılabilir ancak haricî olay zaman aralığının en yüksek sayma değerini (65 535) **aşmamasına** dikkat edilmelidir.

CTC çalışma modunda, ICR1 değeri TOP değeri olarak ayarlanabilir. Bu durumda TCNT1 değeri, ICR1 değerine eşit olduğunda sayıcı sıfırlanır ve karşılaştırma eşleşmesi durumu oluşur. Böylece çıkış karşılaştırma işaretinin frekansını değiştirmek mümkün olur.

4.1.20. Z/S1 PWM Çalışma Modları

Z/S1 modülünün; normal, karşılaştırma eşleşmesinde Z/S'yi sıfırlama, hızlı PWM, doğru faz PWM, doğru faz ve frekans PWM olmak üzere beş adet çalışma modu bulunur. Normal mod ve karşılaştırma eşleşmesinde Z/S'yi sıfırlama modu, Z/S0 modülü ile aynı özelliklere sahip olup Z/S0 modülünden farklı özellikler gösteren PWM çalışma modları aşağıda verilmektedir.

4.1.20.1. Hızlı PWM Modu

Hızlı PWM modu, yüksek frekanslarda PWM dalga şekli üretebilmeye imkân sağlayan moddur. Hızlı PWM modunun çalışma şekli, Z/S0 hızlı PWM modunun çalışma şekli ile aynıdır. Sayıcı, BOTTOM (0x0000) değerinden başlayarak TOP değerine kadar sayma işlemi gerçekleştirir. Bu arada TCNT1 değeri sürekli olarak OCR1x değeri ile karşılaştırılır. Karşılaştırma sonucunda bu değerlerin eşit olması hâlinde, çıkış karşılaştırma pininin lojik durumu değiştirilir. TOP değerine ulaşıldığında sayıcı, BOTTOM değerine tekrar geri döner ve çıkış karşılaştırma pininin lojik durumu değiştirilir.

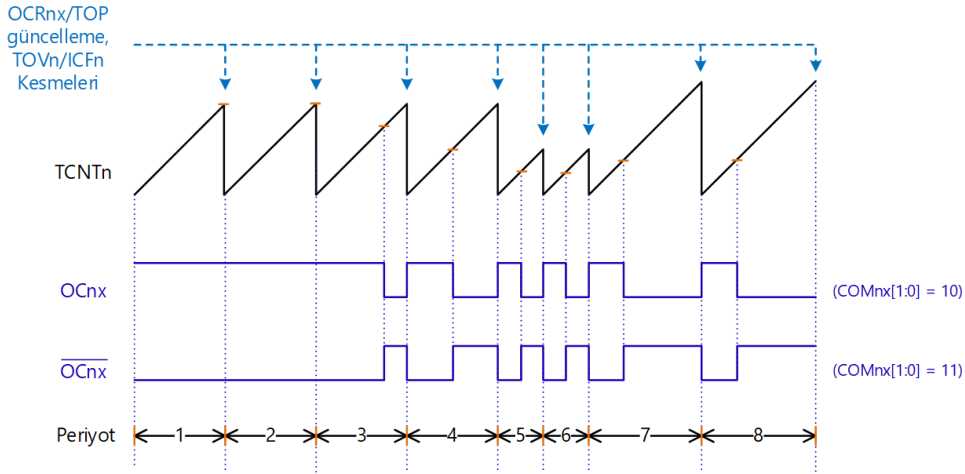
Hızlı PWM'in TOP değeri; 8 bit (0x00FF), 9 bit (0x01FF) veya 10 bit (0x03FF) olarak ayarlanabileceği gibi ICR1 veya OCR1A olarak da ayarlanabilir. ICR1 ya da OCR1A kaydedicilerine MAX (0xFFFF) değeri atanabilir. ICR1 veya OCR1A'ya atanmasına izin verilen en küçük TOP değeri ise 0x0003'tür.

Görsel 4.48’de, Z/S1 modülü hızlı PWM modu için zamanlama diyagramı verilmiştir. Görüldüğü üzere OCR1A, OCR1B, TOP değerlerinin güncellenmesi ve TOV1 ile OCF1 kesmeleri, TOP değerlerinde gerçekleştirilmektedir. Ayrıca sayıcının TOP değeri, farklı değerlere ayarlanabilmekte, üretilen dalga şekli frekansı değiştirilebilmektedir.

Hızlı PWM modunda PWM işaretinin frekansı aşağıdaki eşitlik ile bulunabilir:

$$f_{OCnXPWM} = \frac{f_{CLK_IO}}{N \cdot (1+TOP)}$$

Burada f_{CLK_IO} mikrodenetleyici çalışma saat frekansını ve N ön ölçekleme (prescaler) değerini (1, 8, 64, 256 veya 1024 olabilir) ifade etmektedir.



Görsel 4.48: Hızlı PWM modu zamanlama diyagramı

4.1.20.2. Doğru Faz PWM Modu

Doğru faz PWM modu, yüksek hassasiyette doğru faz PWM dalga şekli üretmek için kullanılan moddur. Doğru faz PWM modunun çalışma şekli, Z/S0 doğru faz PWM modunun çalışma şekli ile aynıdır. Sayıcı, BOTTOM (0x0000) değerinden TOP değerine (ileriye) doğru saydıktan sonra tekrar, BOTTOM değerine (geriye) doğru sayar. Evirmeyen çıkış modunda, sayıcı artarak sayarken TCNT1 ile OCR1x değeri eşit olduğunda OCR1x çıkış durumu sıfırlanır. Sayıcı azalırken eşleşme olduğunda ise OCR1x çıkış durumu 1 olur. Eviren çıkış modunda ise bu işlemin tam tersi gerçekleşir.

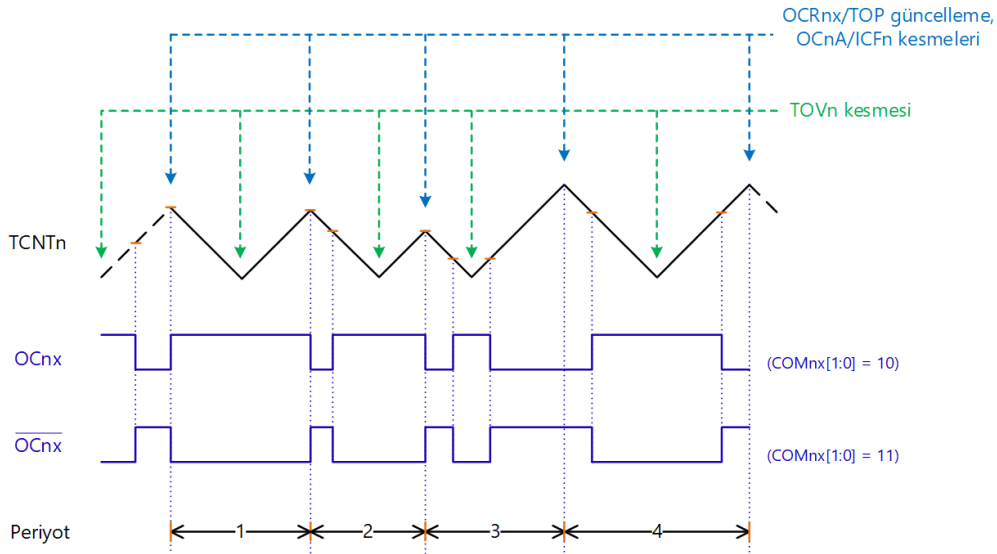
Doğru faz PWM’nin TOP değeri; 8 bit (0x00FF), 9 bit (0x01FF) veya 10 bit (0x03FF) olarak ayarlanabileceği gibi ICR1 ya da OCR1A olarak da ayarlanabilir. ICR1 ya da OCR1A kaydedicilerine MAX (0xFFFF) değeri atanabilir. ICR1 veya OCR1A’ya atanmasına izin verilen en küçük TOP değeri ise 0x0003’tür.

Görsel 4.49'da, Z/S1 modülü doğru faz PWM modu için zamanlama diyagramı verilmiştir. OCR1A, OCR1B, TOP değerlerinin güncellenmesi ve OC1A ile ICF1 kesmelerinin etkinleştirilmesi, sayıcı TOP değerine ulaştığı anlarda yapılmaktadır. Sayıcı BOTTOM değerindeyken ise TOV1 taşıma kesmesi etkinleştirilir.

Doğru faz PWM için çıkış frekansı aşağıdaki formül kullanılarak hesaplanabilir.

$$f_{OCnx_PCPWM} = \frac{f_{CLK_IO}}{2 \cdot N \cdot TOP}$$

Burada f_{CLK_IO} , mikrodenetleyici çalışma saat frekansını, N ise ön ölçekleme (prescaler) değerini (1, 8, 64, 256 veya 1 024 olabilir) ifade etmektedir.



Görsel 4.49: Doğru faz PWM modu zamanlama diyagramı

4.1.20.3. Doğru Faz ve Frekans PWM Modu

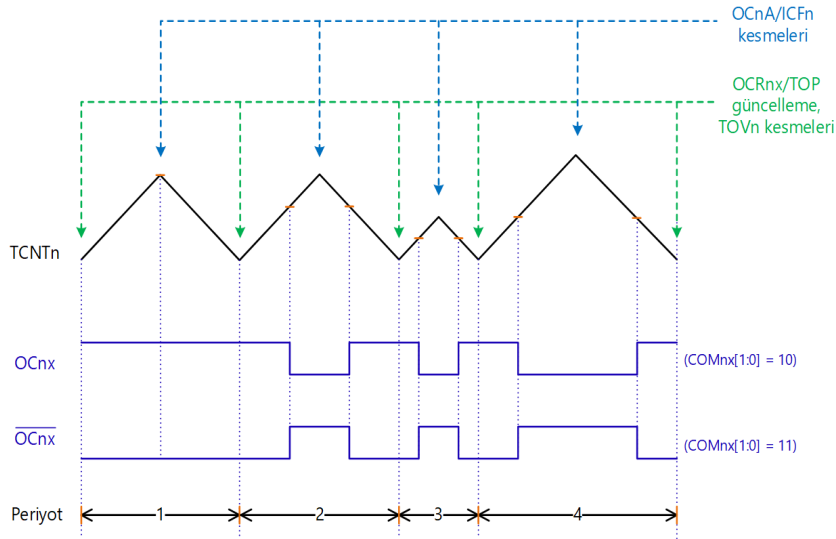
Doğru faz ve frekans PWM modu, yüksek hassasiyette doğru faz ve frekanslı PWM dalga şekli üretmek için kullanılır. Sayıcı sürekli olarak BOTTOM (0x0000) değerinden TOP değerine (ileriye) doğru saydıktan sonra, TOP değerinden tekrar BOTTOM değerine (geriye) doğru sayar. Evirmeyen çıkış karşılaştırma modunda, yukarı doğru sayma işlemi gerçekleştirilirken TCNT1 ve OCR1x değerleri eşitse OC1x pini sıfırlanır, sayıcı aşağı doğru sayarken TCNT1 ve OCR1x değerleri eşit olduğunda ise OC1x pini 1 yapılır. Eviren çıkış karşılaştırma modunda bu işlemin tersi gerçekleştirilir.

Doğru faz ve frekans PWM modunun zamanlama diyagramı Görsel 4.50’de verilmiştir. Doğru faz PWM modundan temel farkı, OCR1x/TOP değerlerinin, sayıcı BOTTOM değerine ulaştığında güncellenmesidir. Böylece, üretilen dalga şeklinde yükselen ve düşen kenarların sağında ve solunda kalan alanlar eşitlenmiş olur. OC1x ve ICF1 bayrakları, sayıcı TOP değerine ulaştığında 1 yapılır. TOV1 kesmesi ise sayıcı BOTTOM değerine ulaşıldığında etkinleştirilir.

Doğru faz ve frekans PWM için çıkış frekansı aşağıdaki formül kullanılarak hesaplanabilir:

$$f_{OCnx_PFCPWM} = \frac{f_{CLK_IO}}{2 \cdot N \cdot TOP}$$

Burada f_{CLK_IO} , mikrodenetleyici çalışma saat frekansını, N ise ön ölçekleme (prescaler) değerini (1, 8, 64, 256 veya 1 024 olabilir) ifade etmektedir.



Görsel 4.50: Doğru faz ve frekans PWM modu zamanlama diyagramı

4.1.21. Z/S1 Modülünde Çıkış Karşılaştırma Modu ve Dalga Şekli Üretme Modunun Belirlenmesi

Z/S1 modülünde çıkış karşılaştırma modu ve dalga şekli üretme modları, Z/S1 kontrol kaydedicisi A (TCCR1A) bitleri ile belirlenmektedir. TCCR1A kaydedicisinin bitleri, Görsel 4.51’de verilmiştir.

	7	6	5	4	3	2	1	0
TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10

Görsel 4.51: TCCR1A bitleri

7-6. Bitler COM1A[1:0] ve 5-4. Bitler COM1B[1:0]: Karşılaştırma eşleşmesi çıkışı A/B modu bitleridir. Bu bitler, OC1x pininin davranışını kontrol eder. OC1x pini, COM1x[1:0]=00 olması hâlinde normal port pini olarak çalışır. Diğer durumlarda ise Z/S'nin çalışma moduna göre farklı işlevler yüklenir.

OC1x, normal veya CTC çalışma modlarında Tablo 4.24'te verilen işlevlere sahiptir.

Tablo 4.24: PWM Olmayan Durumlar İçin Çıkış Karşılaştırma Modu

COM1A1 COM1B1	COM1A0 COM1B0	Tanım
0	0	Normal port çalışması OC1A/OC1B bağlantısı devre dışı.
0	1	Karşılaştırma eşleşmesi hâlinde OC1A/OC1B için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesi hâlinde OC1A/OC1B'yi sıfırla.
1	1	Karşılaştırma eşleşmesi hâlinde OC1A/OC1B'yi 1 yap.



NOT

Karşılaştırma eşleşmesi (compare match), TCNT1 ile OCRA1/OCRB1'in karşılaştırılması sonucunda eşit olması durumudur.

OC1A/OC1B, hızlı PWM modunda Tablo 4.25'te verilen işlevlere sahiptir.

Tablo 4.25: Hızlı PWM Modu İçin Çıkış Karşılaştırma Modu

COM1A1 COM1B1	COM1A0 COM1B0	Tanım
0	0	Normal port çalışması OC0A/OC1B bağlantısı devre dışı.
0	1	WGM1[3:0] değeri 14 ve 15 iken karşılaştırma eşleşmesinde OC1A toggle, OC1B devre dışı (normal port). Diğer WGM1 ayarlarında OC1A ve OC1B devre dışı (normal port).
1	0	Karşılaştırma eşleşmesinde OC1A/OC1B'yi sıfırla, sayıcı BOTTOM (0) değerine ulaştığında OC1A/OC1B'yi 1 yap (evirmeyen mod).
1	1	Karşılaştırma eşleşmesinde OC1A/OC1B'yi 1 yap, sayıcı BOTTOM (0) değerine ulaştığında OC1A/OC1B'yi sıfırla (eviren mod).

OC1A/OC1B, doğru faz PWM modu ile doğru faz ve frekans PWM modu için Tablo 4.26'da verilen işlevlere sahiptir.

Tablo 4.26: Doğru Faz PWM Modu ile Doğru Faz ve Frekans PWM Modu İçin Çıkış Karşılaştırma Modu

COM1A1 COM1B1	COM1A0 COM1B0	Tanım
0	0	Normal port çalışması OC1A/OC1B bağlantısı devre dışı.
0	1	WGM1[3:0] değeri 9 ve 11 iken karşılaştırma eşleşmesinde OC1A toggle, OC1B devre dışı (normal port). Diğer WGM1 ayarlarında OC1A ve OC1B devre dışı (normal port).
1	0	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC1A/OC1B'yi sıfırla. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC1A/OC1B'yi 1 yap.
1	1	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC1A/OC1B'yi 1 yap. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC1A/OC1B'yi sıfırla.

1-0. Bitler WGM11 ve WGM10: Bu bitler, Z/S'nin çalışma modunu ayarlamak için kullanılır. TCCR1B kaydedicisinde bulunan WGM1[3:2] bitleriyle birlikte işlev yaparlar. Bu bitler, sayıcının sayma sürecini kontrol eder, en yüksek (TOP) sayıcı değerini belirler ve nasıl bir dalga şekli üretileceğini ayarlar. WGM13, WGM12, WGM11 ve WGM10 bitlerinin işlevi Tablo 4.27'de verilmiştir (MAX = 0xFFFF, BOTTOM = 0x0000).

Tablo 4.27: Z/S1 Çalışma Modları

Mod	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Çalışma Modunun Adı	TOP	OCR1x Güncelleme	TOV Bayrağı
0	0	0	0	0	Normal	0xFFFF	Derhâl	MAX
1	0	0	0	1	Doğru faz PWM, 8 bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	Doğru faz PWM, 9 bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	Doğru faz PWM, 10 bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Derhâl	MAX
5	0	1	0	1	Hızlı PWM, 8 bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Hızlı PWM, 9 bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Hızlı PWM, 10 bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	Doğru faz ve frekans PWM	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	Doğru faz ve frekans PWM	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	Doğru faz PWM	ICR1	TOP	BOTTOM
11	1	0	1	1	Doğru faz PWM	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Derhâl	MAX
13	1	1	0	1	<i>Rezerve</i>	-	-	-
14	1	1	1	0	Hızlı PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Hızlı PWM	OCR1A	BOTTOM	TOP

4.1.22. Z/S1 Modülünde Giriş Yakalama Ayarları ve Saat Darbesi Kaynağı Seçimi

Z/S1 modülünde giriş yakalama ile ilgili ayarlar ve saat darbesi kaynağının seçimi için Z/S1 kontrol kaydedicisi B (TCCR1B) kullanılır. TCCR1B bitleri, Görsel 4.52’de gösterilmektedir.

	7	6	5	4	3	2	1	0
TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

Görsel 4.52: TCCR1B bitleri

7. Bit ICNC1: Giriş yakalama gürültü engelleyici bitidir. Bu bitin 1 yapılması ile gürültü engelleyici etkinleştirilir. Gürültü engelleyici sayesinde istenmeyen işaret bileşenleri, ICP1 pininden uygulanan işarettten filtrelendir.

6. Bit ICES1: Giriş yakalama kenar seçimi bitidir. Yakalama işleminin hangi kenarda gerçekleştirileceğini belirler. Yakalama işlemi, ICES1 değeri 0 iken düşen kenarda, 1 iken ise yükselen kenarda tetiklenir. Tetikleme işlemi ile TCNT1 değeri, ICR1’e yazılır. Aynı zamanda ICF1 bayrağı 1 yapılır. Kesmeler etkinse giriş yakalama kesmesi çalışır.

2-0. Bitler CS1[2:0]: Z/S1 için saat darbesi seçim bitleridir. Seçme işlemi Tablo 4.28’e göre gerçekleştirilir.

Tablo 4.28: Saat Darbesi Kaynağı Seçimi

CS12	CS11	CS10	Tanım
0	0	0	Saat darbesi kaynağı yok (Zamanlayıcı / sayıcı çalışmaz.).
0	0	1	clkI/O/1 (Ön ölçekleme yok.)
0	1	0	clkI/O/8 (Ön ölçekleyici)
0	1	1	clkI/O/64 (Ön ölçekleyici)
1	0	0	clkI/O/256 (Ön ölçekleyici)
1	0	1	clkI/O/1024 (Ön ölçekleyici)
1	1	0	Haricî saat kaynağı pini [T0 (düşen kenar tetiklemeli)]
1	1	1	Haricî saat kaynağı pini [T0 (yükselen kenar tetiklemeli)]

4.1.23. Z/S1 Modülü Zorlamalı Çıkış Karşılaştırma İşlemi

Z/S1 modülünde zorlamalı çıkış karşılaştırma işlemi, Z/S1 kontrol kaydedicisi C (TCCR1C) üzerinden ayarlanır. TCCR1C bitleri, Görsel 4.53’te verilmiştir.

	7	6	5	4	3	2	1	0
TCCR1C	FOC1A	FOC1B	-	-	-	-	-	-

Görsel 4.53: TCCR1C bitleri

7-6. Bitler FOC1A ve FOC1B: Bu bitler 1 yapıldığında ilgili dalga şekli üretim biriminde otomatik olarak karşılaştırma eşleşmesi durumu oluşturulur. Bu bitler, yalnızca PWM olmayan çalışma modlarında etkindir. COM1x[1:0] bitlerinin durumuna göre OC1A veya OC1B pini çıkışı değiştirilir ancak bu bitlerin değiştirilmesi OC1x kesmesini tetiklemez.

4.1.24. Z/S1 Modülü Z/S Kaydedicileri

Z/S1 modülünde Z/S değeri, her biri 8 bitlik TCNT1H ve TCNT1L kaydedicilerinde saklanır. TCNT1H yüksek düzeyli bitleri (8-15. bitleri), TCNT1L ise düşük düzeyli bitleri (0-7. bitleri) bulundurmaktadır. TCNT1H ve TCNT1L kaydedicilerinin değerleri, TCNT1 değişkeni ile doğrudan okunabilir. TCNT1 kaydedicileri, Görsel 4.54'te verilmiştir.

TCNT1H	TCNT1[15:8]
TCNT1L	TCNT1[7:0]

Görsel 4.54: TCNT1 kaydedicileri

4.1.25. Z/S1 Çıkış Karşılaştırma Kaydedicileri

Z/S1 modülünde yer alan 16 bitlik çıkış karşılaştırma kaydedicileri (OCR1A ve OCR1B), Görsel 4.55'te verilmiştir. OCR1AH ve OCR1BH yüksek düzeyli bitleri (8-15. bitleri), OCR1AL ve OCR1BL ise düşük düzeyli bitleri (0-7. bitleri) bulundurmaktadır. OCR1A ve OCR1B değerleri TCNT1 kaydedicisi ile karşılaştırılarak eşitlik durumunda çıkış karşılaştırma kesmesi üretilir. Ayrıca OC1A ve OC1B çıkış karşılaştırma bitlerindeki lojik durum değiştirilir. OCR1A ve OCR1B değişkenleri kullanılarak bu kaydedicilere doğrudan değer atanabilir veya bu kaydedicilerin değeri okunabilir.

OCR1AH	OCR1A[15:8]
OCR1AL	OCR1A[7:0]
OCR1BH	OCR1B[15:8]
OCR1BL	OCR1B[7:0]

Görsel 4.55: OCR1A ve OCR1B çıkış karşılaştırma kaydedicileri

4.1.26. Z/S1 Modülü Giriş Yakalama Kaydedicileri

Z/S1 modülünde yer alan giriş yakalama kaydedicileri (ICR1A ve ICR1B), Görsel 4.56'da verilmiştir. Giriş yakalama, ICP1 pininden uygulanan dalga şeklinin değişimine göre tetiklenerek TCNT1 değeri ile eşitlenir. Böylece, uygulanan dalga şekli hakkında bilgi edinilebilir. ICR1A ve ICR1B değişkenleri ile bu kaydedicilerin değerleri doğrudan okunabilir.

ICR1AH	ICR1A[15:8]
ICR1AL	ICR1A[7:0]
ICR1BH	ICR1B[15:8]
ICR1BL	ICR1B[7:0]

Görsel 4.56: ICR1A ve ICR1B Giriş yakalama kaydedicileri

4.1.27. Z/S1 Modülü Kesmeleri Etkinleştirme

Z/S1 modülü kesmelerini etkinleştirmek için Z/S kesme maskeleme kaydedicisi (TIMSK1) kullanılır. TIMSK1 bitleri, Görsel 4.57’de verilmiştir.

	7	6	5	4	3	2	1	0
TIMSK1	-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1

Görsel 4.57: TIMSK1 bitleri

5. Bit ICIE1: Giriş yakalama, kesme, etkinleştirme bitidir. SREG kaydedicisi içerisinde yer alan I biti ve ICIE1 biti 1 olduğunda giriş yakalama kesmesi etkin olur.

2. Bit OCIE1B: OCIE1B biti 1 yapıldığında Z/S karşılaştırma eşleşmesi B kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

1. Bit OCIE1A: OCIE1A biti 1 yapıldığında Z/S karşılaştırma eşleşmesi A kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

0. Bit TOIE1: TOIE1 biti 1 yapıldığında Z/S taşma kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

4.1.28. Z/S1 Modülü Kesme Bayraklarının Kontrolü

Z/S1 modülünde yer alan kesme bayraklarının kontrolü için Z/S kesme bayrak kaydedicisi (TIFR1) kullanılır. TIFR1 bitleri, Görsel 4.58’de verilmiştir.

	7	6	5	4	3	2	1	0
TIFR1	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

Görsel 4.58: TIFR1 bitleri

5. Bit ICF1: Giriş yakalama bayrağı bitidir. ICF1 pininde lojik durum değişerek yakalama işlemi tetiklendiğinde ICF1=1 olur. Kesmeler etkinse program, giriş yakalama kesmesi vektörüne dallanır.

2. Bit OCF1B: OCR1B ile TCCR1’in değerleri eşit olduğunda (karşılaştırma eşleşmesi durumunda) karşılaştırma eşleşmesi bayrağı (OCF1B) 1 yapılır. Kesmeler etkinse program, karşılaştırma eşleşmesi B kesme vektörüne dallanır.

1. Bit OCF1A: OCR1A ile TCCR1'in değerleri eşit olduğunda (karşılaştırma eşleşmesi durumunda) karşılaştırma eşleşmesi bayrağı (OCF1A) 1 yapılır. Kesmeler etkinse program, karşılaştırma eşleşmesi A kesme vektörüne dallanır.

0. Bit TOV1: TCCR1 değeri en yüksek değere ulaştıktan sonra, TOV1 taşma bayrağı 1 yapılır. Kesmeler etkinse program, Z/S taşması kesme vektörüne dallanır.

4.1.29. Z/S1 Modülü Çalıştırma Adımları

Z/S1 modülü, seçilen çalışma moduna göre ayarlanır. Kodlama adımları aşağıda sıralanmıştır:

1. Öncelikle Z/S çalışma modu belirlenir. Bunun için TCCR1A ve TCCR1B kaydedicilerinde yer alan WGM1[3:0] bitleri Tablo 4.27'ye göre ayarlanır.

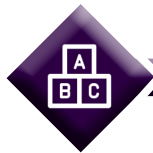
2. Çıkış karşılaştırma pininin işlevi belirlenir. Bunun için TCCR1A kaydedicisinde yer alan COM1A[1:0] bitleri ayarlanır. Bu bitler çalışma moduna göre farklı işlevlere sahiptir. Normal ve CTC çalışma modlarında Tablo 4.24'e göre ayarlama yapılır. Hızlı PWM çalışma modunda, Tablo 4.25'e göre bitler belirlenir. Doğru faz PWM ile doğru faz ve frekans PWM için ise Tablo 4.26'ya göre işlem gerçekleştirilir.

3. Giriş yakalama işlemi yapılacaksa TCCR1B kaydedicisinde yer alan gürültü engelleyici etkinleştirme (ICNC1) ve giriş yakalama kenar seçimi (ICES1) bitleri ayarlanır.

4. ICR1 ve TCNT1 kaydedicileri sıfırlanır. OCR1x karşılaştırma değeri atanır.

5. Kullanılacak kesme kaynakları belirlenir. Bunun için öncelikle SREG kaydedicisinde yer alan I biti 1 yapılır ve global kesmeler etkinleştirilir. Daha sonra TIMSK1 kaydedicisinde yer alan bitler kullanılarak ilgili kesmeler etkin yapılır.

6. Saat darbesi kaynağı ve ön ölçekleme değeri belirlenir. Bunun için TCCR1B kaydedicisinde yer alan CS1[2:0] bitleri, Tablo 4.28'e göre ayarlanır. Bu işlemle birlikte, Z/S çalışmaya başlar.



UYGULAMA

Adı:	Kronometre	No : 4.7
AMAÇ:	Z/S Modülü kullanarak kronometre devresi tasarlamak ve gömülü kodunu yazmak.	Süre: 60 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda Z/S1 modülü ile Z/S değerini, kronometre butonuna basıldığı ve bırakıldığı anları, mili saniye (ms) hassasiyetinde LCD ekranda gösteren, "Sıfırla" butonuna basıldığında sıfırdan başlayan bir kronometre tasarlayınız.

Kullanılacak Araç Gereç: Tablo 4.29'da belirtilmiştir.

Tablo 4.29: Uygulama 4.7 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328P, 28 pinli DIP soket	1 Adet
LCD	16x2 karakter	1 adet
Direnç	1 kΩ	1 adet
Potansiyometre	5 kΩ	1 adet
Breadboard		1 adet
Push buton	İki ayaklı	2 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **Timer_Uyg_03** adlı yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden, **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: Projenizde **main.c** dosyasının bulunduğu dizinde **Include** adında bir dizin oluşturup içerisine 4.2 No.lu uygulamanın 3. adımında verilen **lcd.h** adındaki dosyayı ekleyiniz. Bu başlık dosyasının, LCD'yi kullanabilmeniz için gerekli olduğunu unutmayınız.

4. Adım: **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <stdlib.h>
#include ".\Include\lcd.h"           // lcd.h dosyasını dahil et.

unsigned long int yakalanan = 0;     // Yakalanan değer değişkeni
unsigned int sayac = 0;              // Sayaç değişkeni (zamanlayıcı taşma sayısı)
float saniye = 0, syakalanan = 0;   // Zamanlayıcı ve yakalanan değerler

ISR (TIMER1_CAPT_vect)              // Giriş yakalama kesmesi
{
    yakalanan = ICR1;                // Yakalanan değeri oku.
    syakalanan = yakalanan * 0.001024 + (sayac * 67.108864);
                                     // Yakalanan değeri saniyeye dönüştür.
    ICR1 = 0;                        // Giriş yakalama kaydedicisini sıfırla.
    TCCR1B ^= (1 << ICES1);          // Düşen/yükselen kenar tetikleme (toggle)
}
```



```

ISR (TIMER1_OVF_vect)           // Zamanlayıcı taşma kesmesi
{
    sayac++;                     // Taşma sayısı değişkenini 1 artır.
}

int main(void)
{
    char showsvalue[16], showcvalue[16];
                                // Zamanlayıcı ve yakalanan değerler karakter dizisi

    DDRB = 0;                   // PORTB giriş portu
    PORTB = 0b00000101;         // ICP1 ve PORTB3 pull-up dirençleri etkin.

    LCD_Init();                  // LCD'yi tanı.
    LCD_Clear();                 // LCD'yi temizle.

    sprintf (showsvalue,"%13.5f s", saniye);    // Saniyeyi metne çevir
    sprintf (showcvalue,"%13.5f s", syakalanan); // Yakalanan değeri metne çevir.

    LCD_Printpos(0,0,showsvalue);    // 1. satır 1. sütuna saniye ilk değerini yaz.

    LCD_Printpos(1,0,showcvalue);    // 2. satır 1. sütuna yakalanan değişkeni ilk
                                    // değerini yaz.

    TCCR1A &= ~((1 << WGM11) | (1 << WGM10));    // Normal mod:

    TCCR1B &= ~((1 << WGM13) | (1 << WGM12));    // WGM1[3:0]=0000
    TCCR1A &= ~((1 << COM1A1) | (1 << COM1A0));
                                    // Çıkış karşılaştırma modu A:Devre dışı
    TCCR1A &= ~((1 << COM1B1) | (1 << COM1B0));
                                    // Çıkış karşılaştırma modu B:Devre dışı.

    TCCR1B |= (1 << ICNC1);          // Gürültü engelleyici etkin.
    TCCR1B &= ~((1 << ICES1));        // Düşen kenar tetikleme
    TCCR1C = 0x00;                   // TCCR1C kaydedicisini sıfırla.
    TIMSK1 |= (1 << ICIE1) | (1 << TOIE1);
                                    // Giriş yakalama ve taşma kesmeleri etkin.
    sei();                           // Kesmeler etkin.
    TIFR1 |= (1 << ICF1) | (1 << TOV1);
                                    // Giriş yakalama ve taşma kesme bayraklarını.
                                    // (1 yaparak) temizle.
    TCNT1 = 0;                       // Zamanlayıcı değerini sıfırla.
    TCCR1B |= (1 << CS12) | (1 << CS10);
    TCCR1B &= ~((1 << CS11));        //Saat darbesi, CS1[2:0]=101 Ön ölçekleme  $f_{io}/1024$ 
    while (1)
    {
        saniye = (TCNT1 + 1) * 0.001024 + (sayac * 67.108864);
                                    // Anlık zamanlayıcı değerini saniyeye dönüştür
        sprintf (showsvalue, "%13.5f s", saniye);
        sprintf (showcvalue, "13.5f s", syakalanan);
                                    // Yakalanan değerini metne çevir
        LCD_Printpos(0, 0, showsvalue);    // 1. satır 1. sütuna saniyeyi yaz.
    }
}

```

```

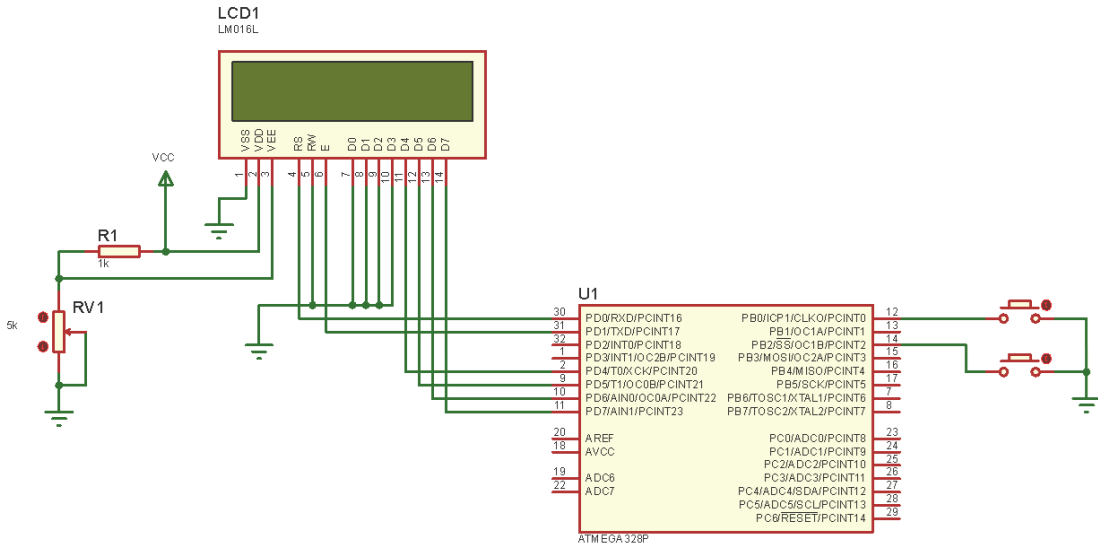
LCD_Printpos(1, 0, showcvalue);
// 2. satır 2. sütuna yakalanan sayıyı yaz.
if ((PINB & (1 << PINB2)) == 0)
// Eğer PORTB2 pinine 0 geldiyse (Sıfırla butonu)
{
    sayac = 0; // Sayacı sıfırla.
    TCNT1 = 0; // Zamanlayıcıyı sıfırla.
    saniye = 0; // Saniyeyi sıfırla.
    sprintf(showsvalue, "%13.5f s", saniye); // Saniyeyi sıfırla.
    sprintf(showsvalue, "%13.5f s", saniye); // Saniyeyi metne çevir.
    LCD_Printpos(0, 0, showsvalue); // Saniyeyi yazdır.
}
_delay_ms(250); // 250 ms bekle.
}
}

```

5. Adım: Kodlar içerisinde kullandığımız **sprintf()** fonksiyonunun doğru çalışabilmesi için 4.2 No.lu uygulamanın 5 ve 6. adımlarında verilen ayarlamaları yapınız.

6. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğunu hatırlayınız.

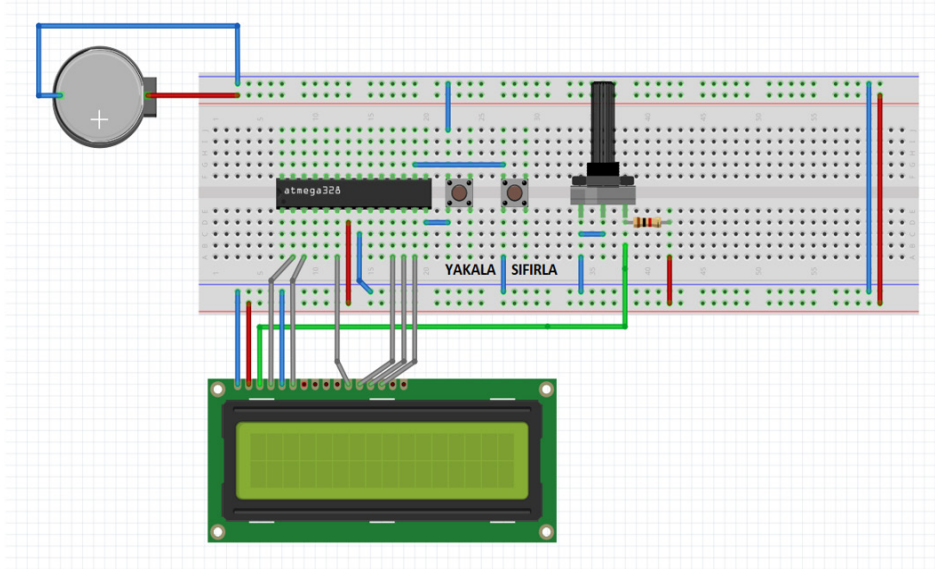
7. Adım: Devre simülasyon programında, Görsel 4.59’da verilen devreyi kurarak simüle ediniz.



Görsel 4.59: Uygulama 4.7 için simülasyon devre şeması

8. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

9. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.60'ta verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.60: Uygulama 4.7 için breadboard üzerindeki devre

10. Adım: “Yakala” ve “Sıfırla” butonlarına basarak devrenin çalışmasını deneyiniz.

11. Adım: Güç kaynağını kapatınız.

12. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Devre simülasyonu başarı ile gerçekleştirildi.		
3. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Yakala ve sıfırla butonları yardımıyla LCD’de gösterilen zaman değeri bir kronometre ile karşılaştırılarak kontrol edildi.		
6. Temizliğe dikkat edildi.		



SIRA SİZDE

4.7 No.lu uygulamada hem butona basıldığı hem de butonun serbest bırakıldığı anlarda yakalama işlemi gerçekleştirilmiştir. Yalnızca butona basıldığı anların yakalanması için gömülü programda hangi değişiklik yapılmalıdır? Bulduğunuz çözümü simülasyon programında simüle ediniz ve devre üzerinde deneyiniz.



UYGULAMA

Adı:	Lambanın Parlaklığını Değiştirme	No : 4.8
AMAÇ:	Z/S Modülü kullanarak lambanın parlaklık kontrolünü yapan devre tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda Z/S1 modülü yardımıyla artır veya azalt butonlarına basıldığında lambanın parlaklığını artıran veya azaltan mikrodenetleyici devresini tasarlayınız ve gömülü kodunu yazınız.

PB0 pinini **Azalt**, PB2 pinini ise **Artır** butonu için kullanınız.

Kullanılacak Araç Gereç: Tablo 4.30'da belirtilmiştir.

Tablo 4.30: Uygulama 4.8 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328, 28 pinli DIP soket	1 adet
Direnç	150 Ω	1 adet
Mini akkor ampul	2.5 V	1 adet
Duy	Mini akkor ampul için	1 adet
NPN transistor	2N3904	1 adet
Breadboard		1 adet
Push buton	İki ayaklı	2 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

- 1. Adım:** 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **Timer_Uyg_04** adlı yeni bir proje oluşturunuz.
- 2. Adım:** 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden, **Atmega328P** mikrodenetleyicisini seçiniz.
- 3. Adım:** **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```
#include <avr/io.h>
```

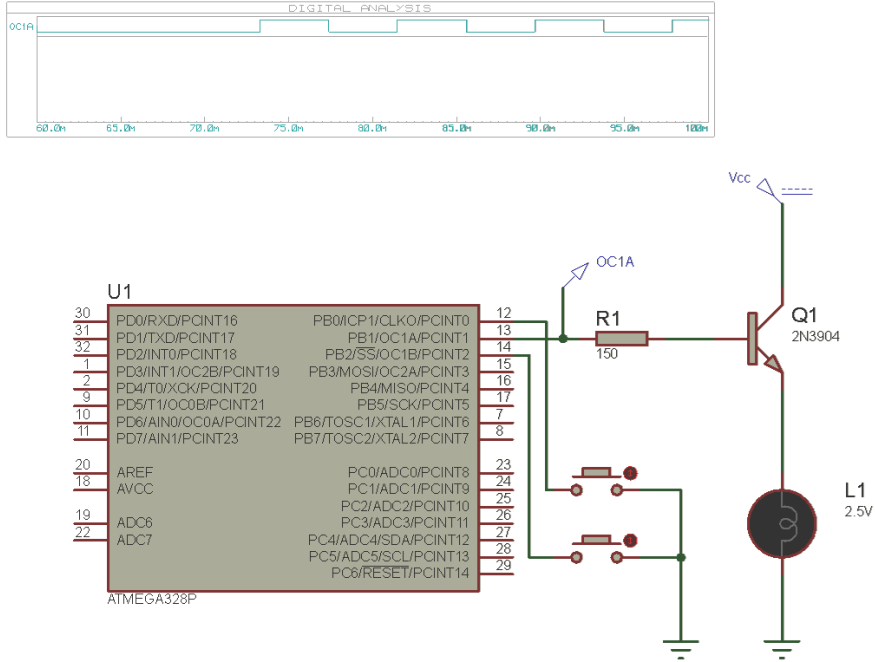
```
#include <avr/interrupt.h>
```

```
ISR (TIMER1_COMPA_vect)                                // Çıkış karşılaştırma kesmesi
{
    if(((PINB & (1 << PINB0)) == 0) && (OCR1A > 5))      // PORTB 0. pin basılıysa
        OCR1A = OCR1A - 5;                             // OCR1A'yı 5 azalt.
    if(((PINB & (1 << PINB2)) == 0) && (OCR1A < 1018))   // PORTB 2. pin basılıysa
        OCR1A = OCR1A + 5;                             // OCR1A'yı 5 artır.
}

int main(void)
{
    PORTB = 0b00000101;                                // PORTB0 ve PORTB2 için pull-up dirençler etkin.
    DDRB = 0b00000010;                                // OC1A (PORTB1) pini çıkış
    TCCR1B &= ~(1 << WGM13);                            // Hızlı PWM 10 bit:
    TCCR1B |= (1 << WGM12);                             // WGM1[3:0]=0111
    TCCR1A |= (1 << WGM11) | (1 << WGM10);
    TCCR1A |= (1 << COM1A1);                            // Evirmeyen mod:
    TCCR1A &= ~(1 << COM1A0);                          // COM1A[1:0]=10
    TCCR1C = 0;                                         // TCCR1C sıfırla.
    TCNT1 = 0;                                         // Zamanlayıcıyı sıfırla.
    OCR1A = 512;                                       // Çıkış karşılaştırma ilk değerini ayarla.
    TIMSK1 |= (1 << OCIE1A);                          // Çıkış karşılaştırma kesmesi etkin.
    TIFR1 |= (1 << OCF1A);                             // Çıkış karşılaştırma bayrağını temizle.
    TCCR1B |= (1 << CS11);                             // Saat darbesi modu:
    TCCR1B &= ~((1 << CS12) | (1 << CS10));
                                                    // CS1[2:0]=010 clk_I/O / 8 ön ölçekleme
    sei();                                           // Kesmeler etkin.
    while (1);                                       // Sonsuz döngü
}
```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğunu hatırlayınız.

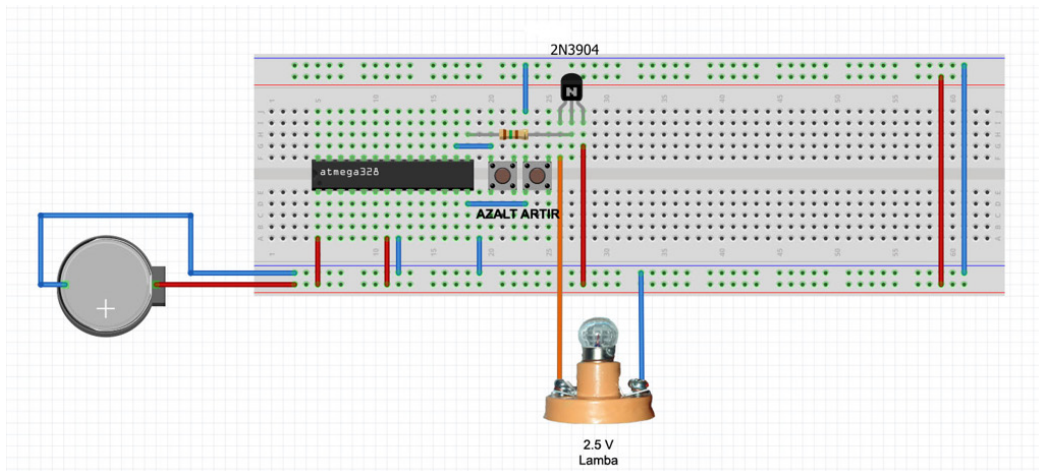
5. Adım: Devre simülasyon programında, Görsel 4.61'de verilen devreyi kurarak simüle ediniz.



Görsel 4.61: Uygulama 4.8 için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.62'de verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.62: Uygulama 4.8 için breadboard üzerindeki devre

8. Adım: Azalt ve artır butonlarına basarak devrenin çalışmasını deneyiniz.

9. Adım: Güç kaynağını kapatınız.

10. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Devre simülasyonu başarı ile gerçekleştirildi.		
3. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Azalt ve artır butonlarına basılarak ışık parlaklığının değişimi tespit edildi.		
6. Temizliğe dikkat edildi.		



NOT

Kaynak gerilimi $V_{cc} = 5\text{ V}$ iken mikrodenetleyici pini üzerinden çekilebilecek en yüksek kaynak (source) akımı, 20 mA civarındadır. Mini akkor ampul, daha fazla akıma ihtiyaç duyacağından doğrudan mikrodenetleyici pinine bağlanarak ışık vermez. Akımı yükseltmek amacıyla devrede bir NPN transistör (2N3904) kullanılmıştır. Böylece lamba için 200 mA değerine kadar akım çekilerek lambanın yeterli parlaklıkta yanması sağlanır.

4.1.30. Z/S2 Modülü

Z/S2, tek kanallı 8 bitlik bir Z/S modülüdür. Z/S (TCNT2) ve çıkış karşılaştırma kaydedicileri (OCR2A ve OCR2B), 8 bitlik kaydedicilerdir. Kesmeler, Z/S kesme bayrağı kaydedicisinde (TIFR2) bulunan bitlerle takip edilir. Kesmeleri etkinleştirmek için zamanlayıcı kesme maskeleye kaydedicisi (TIMSK2) kullanılır. Saat darbesi kaynağı olarak dâhilî saat darbesi, ön ölçekleme veya haricî saat darbesi seçilebilir.

Çıkış karşılaştırma kaydedicileri (OCR2A ve OCR2B), sürekli olarak TCNT2 değeri ile karşılaştırılır. Karşılaştırma sonucuna göre dalga şekli üretici, çıkış karşılaştırma pinleri (OC2A ve OC2B) üzerinden PWM veya değişik frekansta çıkış dalga şekilleri üretebilir.

Z/S2, 8 bitlik bir Z/S olduğundan 0 ile 255 arasında sayabilir. Z/S2 modülü; normal, CTC ve PWM modlarında çalışabilmektedir. Bu modların çalışma prensibi, Z/S0 modülündeki çalışma modları ile aynıdır.

Z/S2; Z/S kontrol kaydedicileri (TCCR2A ve TCCR2B), Z/S (TCNT2), çıkış karşılaştırma kaydedicileri (OCR2A ve OCR2B), Z/S kesme maskeleme kaydedicisi (TIMSK2), Z/S bayrak kaydedicisi (TIFR2) ve asenkron durum kaydedicisi (ASSR) adlı kaydedicilere sahiptir. ASSR kaydedicisi hariç bu kaydediciler, Z/S0 modülü içerisindeki kaydediciler ile benzer bitlere sahiptir.

4.1.31. Z/S2 Modülünde Karşılaştırma Çıkış Modu ve Çalışma Modunun Belirlenmesi

Z/S2 modülü için karşılaştırma çıkış modu ve çalışma modunun belirlenmesi işlemi, Z/S kontrol kaydedicisi A (TCCR2A) üzerinden gerçekleştirilir. TCCR2A kaydedicisinin bitleri, Görsel 4.63'te verilmiştir.

	7	6	5	4	3	2	1	0
TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20

Görsel 4.63: TCCR2A bitleri

7.-6. Bitler COM2A[1:0]: Karşılaştırma eşleşmesi çıkışı A (OC2A) modu bitleridir. Bu bitler, OC2A pininin davranışını kontrol eder. OC2A pini, COM2A[1:0]=00 olması hâlinde normal port pini olarak çalışır. Diğer durumlarda ise Z/S'nin çalışma moduna göre farklı işlevler yüklenir.

OC2A, normal veya CTC çalışma modlarında Tablo 4.31'de verilen işlevlere sahiptir.

Tablo 4.31: PWM Olmayan Durumlar İçin Çıkış Karşılaştırma Modu

COM2A1	COM2A0	Tanım
0	0	Normal port çalışması OC2A bağlantısı devre dışı.
0	1	Karşılaştırma eşleşmesi hâlinde OC2A için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesi hâlinde OC2A'yı sıfırla.
1	1	Karşılaştırma eşleşmesi hâlinde OC2A'yı 1 yap.



NOT

Karşılaştırma eşleşmesi, TCNT2 ile OCRA2'nin karşılaştırılması sonucunda eşit olması durumudur.

OC2A, hızlı PWM modunda Tablo 4.32’de verilen işlevlere sahiptir.

Tablo 4.32: Hızlı PWM Modu İçin Çıkış Karşılaştırma Modu

COM2A1	COM2A0	Tanım
0	0	Normal port çalışması OC2A bağlantısı devre dışı.
0	1	WGM22=0 iken normal port çalışması OC2A bağlantısı devre dışı. WGM22=1 iken karşılaştırma eşlemesinde OC2A için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesinde OC2A’yı sıfırla, sayıcı BOTTOM (0) değerine ulaştığında OC2A’yı 1 yap (evirmeyen mod).
1	1	Karşılaştırma eşleşmesinde OC2A’yı 1 yap, sayıcı BOTTOM (0) değerine ulaştığında OC2A’yı sıfırla (eviren mod).

OC2A, doğru faz PWM modunda Tablo 4.33’te verilen işlevlere sahiptir.

Tablo 4.33: Doğru Faz PWM Modu İçin Çıkış Karşılaştırma Modu

COM2A1	COM2A0	Tanım
0	0	Normal port çalışması OC2A bağlantısı devre dışı.
0	1	WGM22=0 iken normal port çalışması OC2A bağlantısı devre dışı. WGM22=1 iken karşılaştırma eşlemesinde OC2A için toggle işlemi yap.
1	0	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC2A’yı sıfırla. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC2A’yı 1 yap.
1	1	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC2A’yı 1 yap. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC2A’yı sıfırla.

5.-4. Bitler COM2B[1:0]: Karşılaştırma eşleşmesi çıkışı B (OC2B) modu bitleridir. Bu bitler, OC2B pininin davranışını kontrol eder. OC2B pini, COM2B[1:0]=00 olması hâlinde normal port pini olarak çalışır. Diğer durumlarda ise Z/S’nin çalışma moduna göre farklı işlevler yüklenir.

OC2B, normal veya CTC çalışma modlarında Tablo 4.34’te verilen işlevlere sahiptir.

Tablo 4.34: PWM Olmayan Durumlar İçin Çıkış Karşılaştırma Modu

COM2B1	COM2B0	Tanım
0	0	Normal port çalışması OC2B bağlantısı devre dışı.
0	1	Karşılaştırma eşleşmesi hâlinde OC2B için toggle işlemi yap.
1	0	Karşılaştırma eşleşmesi hâlinde OC2B’yi sıfırla.
1	1	Karşılaştırma eşleşmesi hâlinde OC2B’yi 1 yap.

OC2B, hızlı PWM modunda Tablo 4.35'te verilen işlemlere sahiptir.

Tablo 4.35: Hızlı PWM Modu İçin Çıkış Karşılaştırma Modu

COM2B1	COM2B0	Tanım
0	0	Normal port çalışması OC2B bağlantısı devre dışı.
0	1	<i>Rezerve</i>
1	0	Karşılaştırma eşleşmesinde OC2B'yi sıfırla, sayıcı BOTTOM (0) değerine ulaştığında OC2B'yi 1 yap (evirmeyen mod).
1	1	Karşılaştırma eşleşmesinde OC2B'yi 1 yap, sayıcı BOTTOM (0) değerine ulaştığında OC2B'yi sıfırla (eviren mod).

OC2B, doğru faz PWM modunda Tablo 4.36'da verilen işlemlere sahiptir.

Tablo 4.36: Doğru Faz PWM Modu İçin Çıkış Karşılaştırma Modu

COM2B1	COM2B0	Tanım
0	0	Normal port çalışması OC2B bağlantısı devre dışı.
0	1	<i>Rezerve</i>
1	0	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC2B'yi sıfırla. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC2B'yi 1 yap.
1	1	Yukarı doğru sayarken karşılaştırma eşleşmesinde OC2B'yi 1 yap. Aşağı doğru sayarken karşılaştırma eşleşmesinde OC2B'yi sıfırla.

1.-0. Bitler WGM21 ve WGM20: Bu bitler, Z/S'nin çalışma modunu ayarlamak için kullanılır. TCCR2B kaydedicisinde bulunan WGM22 bitiyle birlikte işlev yaparlar. Bu bitler sayıcının sayma sürecini kontrol eder, en yüksek (TOP) sayıcı değerini belirler ve nasıl bir dalga şekli üretileceğini ayarlar. WGM22, WGM21 ve WGM20 bitlerinin işlevi Tablo 4.37'de verilmiştir (MAX = 0xFF, BOTTOM = 0x00).

Tablo 4.37: Z/S2 Çalışma Modları

Mod	WGM22	WGM21	WGM20	Çalışma Modunun Adı	TOP	OCR2x Güncelleme	TOV Bayrağı
0	0	0	0	Normal	0xFF	Derhâl	MAX
1	0	0	1	Doğru faz PWM	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Derhâl	MAX
3	0	1	1	Hızlı PWM	0xFF	BOTTOM	MAX
4	1	0	0	<i>Rezerve</i>	-	-	-
5	1	0	1	Doğru faz PWM	OCRA	TOP	BOTTOM
6	1	1	0	<i>Rezerve</i>	-	-	-
7	1	1	1	Hızlı PWM	OCRA	BOTTOM	TOP

4.1.32. Z/S2 Modülünde Zorlamalı Çıkış Karşılaştırma ve Çalışma Saat Darbesi Seçimi

Normal veya CTC çalışma modlarında, OC2A veya OC2B’de üretilen dalga şeklinin zorlamalı olarak değiştirilmesi ve Z/S çalışma saat darbesi seçimi için Z/S kontrol kaydedicisi B (TCCR2B) kullanılır. TCCR0B bitleri, Görsel 4.64’te verilmiştir.

	7	6	5	4	3	2	1	0
TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20

Görsel 4.64: TCCR2B bitleri

7.-6. Bitler FOC2A ve FOC2B: Bu bitler 1 yapıldığında ilgili dalga şekli üretim biriminde, karşılaştırma eşleşmesi durumu otomatik olarak oluşturulur. COM2x[1:0] bitlerinin durumuna göre OC2A veya OC2B pini çıkışı değiştirilir ancak bu bitlerin değiştirilmesi OC2x kesmesini tetiklemez.

2.-0. Bitler CS2[2:0]: Bu bitler, çalışma saat seçimi bitleridir. Z/S için saat darbesi kaynağını ve ön ölçekleyici değerini belirler. CS2 bitlerinin anlamları, Tablo 4.38’de verilmiştir.

Tablo 4.38: Saat Darbesi Kaynağı Seçimi

CS22	CS21	CS20	Tanım
0	0	0	Saat darbesi kaynağı yok (Z/S çalışmaz.).
0	0	1	clk_{T2S} (Ön ölçekleme yok.)
0	1	0	$clk_{T2S}/8$ (Ön ölçekleyici)
0	1	1	$clk_{T2S}/32$ (Ön ölçekleyici)
1	0	0	$clk_{T2S}/64$ (Ön ölçekleyici)
1	0	1	$clk_{T2S}/128$ (Ön ölçekleyici)
1	1	0	$clk_{T2S}/256$ (Ön ölçekleyici)
1	1	1	$clk_{T2S}/1024$ (Ön ölçekleyici)

Z/S2 saat darbesi kaynağı clk_{T2S} olarak adlandırılır. clk_{T2S} varsayılan olarak ana sistem I/O saat darbesine ($clk_{I/O}$) bağlıdır ancak ASSR kaydedicisinde bulunan AS2 biti 1 yapıldığında Z/S2, TOSC1 pini üzerinden asenkron olarak tetiklenebilir. Böylece gerçek zamanlı bir sayıcı elde edilebilir. TOSC1 ve TOSC2 pinleri tetikleme amacıyla kullanıldığında ilgili PORTB pinleri devre dışıdır. TOSC1 ve TOSC2 pinlerine sistem saat darbesinden bağımsız olmak üzere haricî bir kristal de bağlanabilir.

4.1.33. Z/S2 Modülünde Z/S Değeri

Z/S değeri, 8 bitlik TCNT2 kaydedicisi içerisinde yer alır. Çalışma anında sayıcı, her bir Z/S saat

darbesinde TCNT2'nin değerini 1 artırır veya azaltır. TCNT2'nin değeri, karşılaştırma eşleşmesi anında değiştirilemez, diğer durumlarda değiştirilebilir ancak TCNT2'nin Z/S çalışırken değiştirilmesi karşılaştırma eşleşmesinin kaçırılmasına neden olabileceğinden önerilmez.

4.1.34. Z/S2 Modülünde Çıkış Karşılaştırma Kaydedicileri

Çıkış karşılaştırma (output compare) kaydedicileri (OCR2A ve OCR2B), TCNT2 sayıcı değeri ile karşılaştırılacak değeri tutan 8 bitlik kaydedicilerdir. Karşılaştırma eşleşmesi durumunda, çıkış karşılaştırma kesmesi oluşur ya da OC2A veya OC2B pinlerinde üretilen dalga şekli değişir.

4.1.35. Z/S2 Modülünde Z/S Kesmelerinin Etkinleştirilmesi

Z/S kesmelerinin etkin yapılması için Z/S kesme maskeleme kaydedicisinin (TIMSK2) bitleri ayarlanır. TIMSK2 bitleri, Görsel 4.65'te verilmiştir.

	7	6	5	4	3	2	1	0
TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2

Görsel 4.65: TIMSK2 bitleri

2. Bit OCIE2B: OCIE2B biti 1 yapıldığında, Z/S karşılaştırma eşleşmesi B kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

1. Bit OCIE2A: OCIE2A biti 1 yapıldığında Z/S karşılaştırma eşleşmesi, A kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

0. Bit TOIE2: TOIE2 biti 1 yapıldığında Z/S taşma kesmesi etkinleştirilir. Kesmenin çalışabilmesi için öncelikle SREG kaydedicisinde yer alan I bitinin 1 yapılması gerekir.

4.1.36. Z/S2 Modülünde Z/S Kesme Bayraklarının Kontrolü

Z/S kesmelerinden biri olduğunda ilgili kesme bayrağı biti 1 olur. Bu bayrak biti, kesme vektörüne gidildiğinde donanım tarafından otomatik olarak sıfırlanır. Yazılım içerisinde kesme bayrak bitlerini sıfırlamak için bu bitlere 1 yazılmalıdır. Z/S kesme bayrak bitleri TIFR2 kaydedicisinden kontrol edilir. Görsel 4.66'da TIFR0 bitleri verilmiştir.

	7	6	5	4	3	2	1	0
TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2

Görsel 4.66: TIFR2 bitleri

2. Bit OCF2B: OCR2B ile TCCR2'nin değerleri eşit olduğunda (karşılaştırma eşleşmesi durumunda) karşılaştırma eşleşmesi bayrağı (OCF2B) 1 yapılır.

1. Bit OCF2A: OCR2A ile TCCR2'nin değerleri eşit olduğunda (karşılaştırma eşleşmesi durumunda) karşılaştırma eşleşmesi bayrağı (OCF2A) 1 yapılır.

0. Bit TOV2: TCCR2 değeri en yüksek değere ulaştıktan sonra, taşma bayrağı TOV2 1 yapılır. Kescmeler etkinse zamanlayıcı taşma kesme vektörüne gidilir.

4.1.37. Z/S2 Modülü Çalıştırma Adımları

Z/S2 modülü, seçilen çalışma moduna göre ayarlanır. Kodlama adımları aşağıda sıralanmıştır:

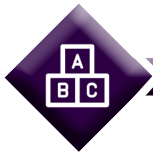
1. Öncelikle Z/S çalışma modu belirlenir. Bunun için TCCR2A ve TCCR2B kaydedicilerinde yer alan WGM2[2:0] bitleri Tablo 4.37'ye göre ayarlanır.

2. Çıkış karşılaştırma pininin işlevi belirlenir. Bunun için TCCR2A kaydedicisinde yer alan COM2A[1:0] bitleri ayarlanır. Bu bitler çalışma moduna göre farklı işlevlere sahiptir. Normal ve CTC çalışma modlarında Tablo 4.31 ve 4.34'e göre ayarlama yapılır. Hızlı PWM çalışma modunda, Tablo 4.32 ve 4.35'e göre bitler belirlenir. Doğru faz PWM için ise Tablo 4.33 ve 4.36'ya göre işlem gerçekleştirilir.

3. TCNT2 kaydedicisi sıfırlanır. OCR2x kaydedicisine karşılaştırma değeri atanır.

4. Kullanılacak kesme kaynakları belirlenir. Bunun için öncelikle *sei()* komutu ile SREG kaydedicisinde I biti 1 yapılarak global kesmeler etkinleştirilir. Daha sonra TIMSK2 kaydedicisinde yer alan bitler kullanılarak ilgili kesmeler etkin yapılır.

5. Saat darbesi kaynağı ve ön ölçekleme değeri belirlenir. Bunun için TCCR2B kaydedicisinde yer alan ve Tablo 4.38'de verilen CS2[2:0] bitleri ayarlanır. Bu işlemle birlikte, Z/S çalışmaya başlar.



UYGULAMA

Adı:	DC Motor Hız Kontrolü	No : 4.9
AMAÇ:	Z/S Modülü kullanarak motorun devir kontrolünü yapan devreyi tasarlamak ve gömülü kodunu yazmak.	Süre: 40 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda, Z/S2 modülü ile PWM dalga şekli üreterek **Artır** butonuna basıldığında DC motor devir sayısını artıran; **Azalt** butonuna basıldığında DC motor devir sayısını azaltan devreyi tasarlayınız ve gömülü yazılımını kodlayınız. PB0 pinini **Azalt**, PB2 pinini ise **Artır** butonu için kullanınız.

Kullanılacak Araç Gereç: Tablo 4.39'da belirtilmiştir.

Tablo 4.39: Uygulama 4.9 için malzeme listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328, 28 pinli DIP soket	1 adet
Direnç	150 Ω	1 adet
DC motor	5 V	1 adet
Diyot	1N4007	1 adet
NPN transistor	2N2222	1 adet
Breadboard		1 adet
Push buton	İki ayaklı	2 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

- 1. Adım:** 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **Timer_Uyg_05** adlı yeni bir proje oluşturunuz.
- 2. Adım:** 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden, **Atmega328P** mikrodenetleyicisini seçiniz.
- 3. Adım:** **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR(TIMER2_COMPA_vect) // Çıkış karşılaştırma kesmesi
{
    if(((PINB & (1 << PINB0)) == 0) && (OCR2A > 0)) // PORTB 0. pin basılıysa
        OCR2A = OCR2A - 1; // OCR2A'yı 1 azalt.
    if(((PINB & (1 << PINB2)) == 0) && (OCR2A < 255)) // PORTB 2. pin basılıysa
        OCR2A = OCR2A + 1; // OCR2A'yı 1 artır.
}

int main(void)
{
    PORTB = 0b00000101; // PORTB0 ve PORTB2 için pull-up dirençler etkin.
    DDRB = 0b000001000; // OC2A (PORTB3) pini çıkış

    TCCR2B &= ~(1 << WGM22); // Zamanlayıcı modu seçimi:
    TCCR2A |= (1 << WGM21) | (1 << WGM20); // WGM2[2:0]=011 Hızlı PWM

    TCCR2A |= (1 << COM2A1); // Çıkış karşılaştırma modu:
    TCCR2A &= ~(1 << COM2A0); // COM2A[1:0]=10, Evirmeyen mod
}
```

```

TCCR2A &= ~((1 << COM2B1) | (1 << COM2B0));
// COM2B etkin değil.

TCNT2 = 0; // Zamanlayıcıyı sıfırla.
OCR2A = 128; // Çıkış karşılaştırma ilk değeri ata.

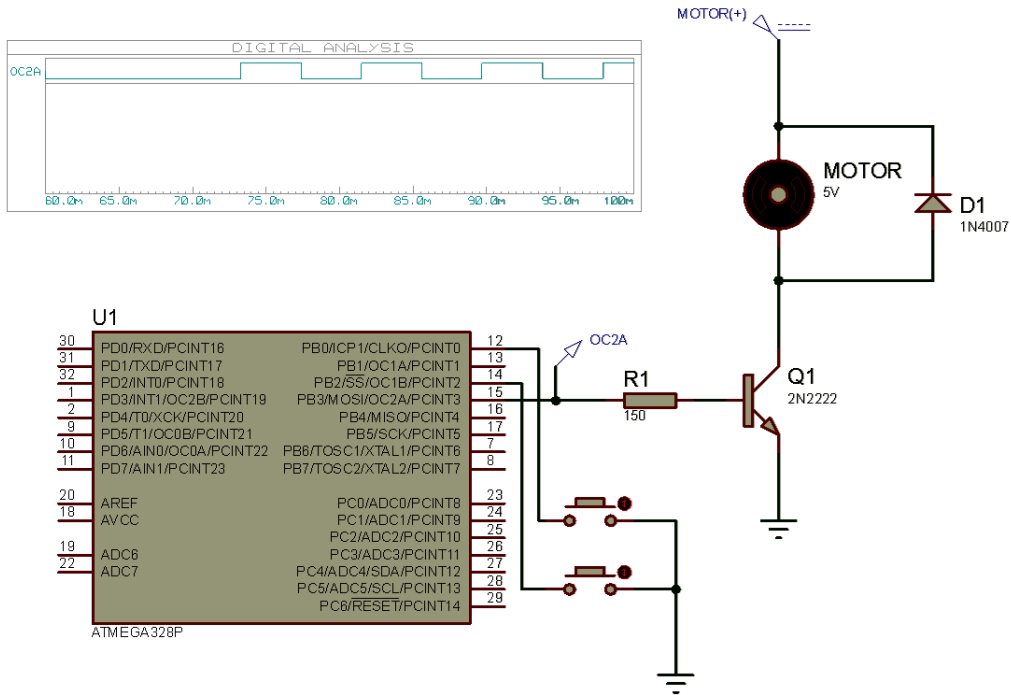
sei(); // Kesmeler etkin.
TIMSK2 |= (1 << OCIE2A); // Çıkış karşılaştırma kesmesi etkin.
TIFR2 = 0xFF; // Tüm kesme bayraklarını sıfırla.
TCCR2B &= ~(1 << CS22); // Saat darbesi kaynağı seçimi:
TCCR2B |= (1 << CS21) | (1 << CS20);
// CS2[2:0]=011 clkT2S/32 ön ölçekleme

while (1); // Sonsuz döngü
}

```

4. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

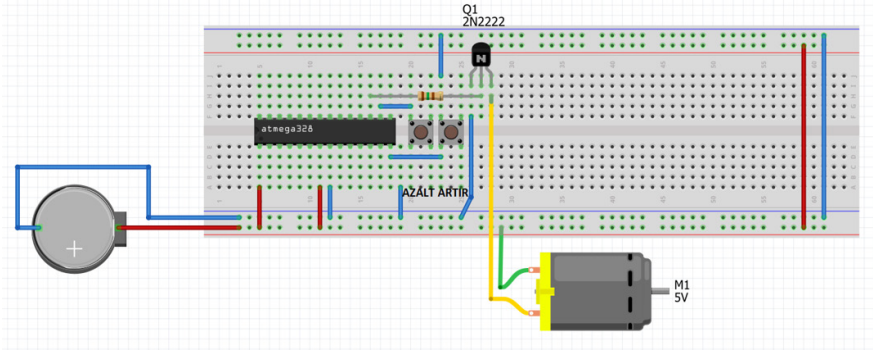
5. Adım: Devre simülasyon programında, Görsel 4.67’de verilen devreyi kurarak simüle ediniz.



Görsel 4.67: Uygulama 4.9 için simülasyon devre şeması

6. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

7. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.68’de gösterilen şekilde kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.68: Uygulama 10 için breadboard üzerindeki devre

8. Adım: Azalt ve Artır butonlarına basarak devrenin çalışmasını deneyiniz. Motor hızının değişimini izleyiniz.

9. Adım: Güç kaynağını kapatınız.

10. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Devre simülasyonu başarı ile gerçekleştirildi.		
3. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Azalt ve Artır butonlarına basılarak DC motor hızının değişimi tespit edildi.		
6. Temizliğe dikkat edildi.		

4.2. MİKRODENETLEYİCİ İLE SICAKLIK KONTROLÜ

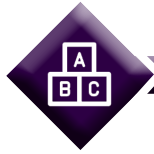
Atmega328P mikrodenetleyici, sıcaklık ölçümü için ADC modülü içerisinde yer alan bir dâhilî algılayıcıya sahiptir. Ayrıca, mikrodenetleyici ile haricî bir sıcaklık algılayıcısı kullanılarak da sıcaklık ölçümü gerçekleştirilebilir.

4.2.1. Mikrodenetleyici Dâhilî Sıcaklık Algılayıcısı

Atmega328P mikrodenetleyicide yer alan ADC modülü içerisinde 1 mV/°C duyarlılık değerine ve ± 10 °C doğruluğa sahip dâhilî bir sıcaklık algılayıcı bulunur. Bu algılayıcı, genellikle mikrodenetleyici kullanılan devrelerin aşırı ısınma durumlarının algılanması amacıyla kullanılır.

ADC modülünde yer alan dâhilî sıcaklık algılayıcısını etkinleştirmek için ADMUX kaydedicisinin MUX[3:0] bitlerini “1000” şeklinde ayarlayarak ADC8 kanalını seçmek gereklidir. Ayrıca ADMUX kaydedicisinin REFS[1:0] bitlerini “11” yaparak dâhilî 1,1 V referans gerilimini seçmek gereklidir. Sıcaklık algılayıcı etkinleştirildiğinde, ADC, sıcaklık algılayıcının üzerindeki gerilimi ölçerek değer üretir. Ölçülen gerilim, sıcaklık değeriyle doğrusal bir ilişkiye sahiptir.

Ölçülen değer, bir mikrodenetleyici entegresinden diğerine göre farklılık gösterebileceğinden, ölçüm değerinin hesaplanması için yazılımda kalibrasyon yapılması gereklidir. 4.10 No.lu uygulamada, dâhilî sıcaklık algılayıcı kullanılarak bir dijital termometre tasarımı yapılmıştır.



UYGULAMA

Adı:	Dâhilî Sıcaklık Algılayıcı ile Sıcaklık Ölçümü	No : 4.10
AMAÇ:	Dâhilî sıcaklık algılayıcıyı kullanabilmek.	Süre: 60 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek diğer sayfada verilen adımlar doğrultusunda, Atmega328P mikrodenetleyicinin dâhilî sıcaklık algılayıcısı ile sıcaklık ölçümü gerçekleştiriniz.

Kullanılacak Araç Gereç: Tablo 4.40’te belirtilmiştir.

Tablo 4.40: Uygulama 4.9 için malzeme listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici	Atmega328, 28 pinli DIP soket	1 adet
LCD	16x2 karakter	1 adet
Direnç	1 kΩ	1 adet
Kondansatör	100 nF	1 adet
Potansiyometre	5 kΩ	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

1. Adım: 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **ADC_Uyg_01** adlı yeni bir proje oluşturunuz.

2. Adım: 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.

3. Adım: Projenizde **main.c** dosyasının bulunduğu dizinde **Include** adında bir dizin oluşturup içerisine 4.2 No.lu uygulamanın 3. adımında verilen **lcd.h** adındaki dosyayı ekleyiniz. Bu başlık dosyasının LCD'yi kullanabilmeniz için gerekli olduğunu hatırlayınız.

4. Adım: **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```
#include <avr/io.h>
#include <stdio.h>
#include <stdio.h>
#include <stdlib.h>
#include "..\Include\lcd.h"           // lcd.h dosyasını dahil et.

int main(void)
{
    volatile int cvalue;              // ADC çevirme sonucunun kaydedildiği değişken
    volatile float tempC;             // Sıcaklık değerinin kaydedildiği değişken
    char showcvalue [16];            // ADC çevirme sonucu karakter dizisi
    char showtvalue [16];            // Gerilim değeri karakter dizisi
    LCD_Init();                      // LCD'yi tanı.
    ADMUX |= ((1 << REFS1) | (1 << REFS0)); // Referans gerilimi seçimi:
                                         // REFS[1:0]=11 Dâhilî 1.1V ref. gerilimi
    ADMUX &= ~(1 << ADLAR);           // Sonuç hizalama (ADLAR=0: sağa yaslı)
    ADMUX |= (1 << MUX3); // ADC giriş kanalı seçimi:
    ADMUX &= ~((1 << MUX2) | (1 << MUX1) | (1 << MUX0));
                                         // (MUX[3:0]=1000: ADC8 sıcaklık algılayıcı)
    ADCSRA |= (1 << ADEN); // ADC etkin (ADEN=1).
    ADCSRA &= ~(1 << ADATE); // Otomatik tetikleme devre dışı (ADATE=0).
    ADCSRA |= ((1 << ADPS2) | (1 << ADPS1));
    ADCSRA &= ~(1 << ADPS0); // Prescaler (Ön ölçekleyici) (ADPS[2:0]=110: /64)

    while (1)
    {
        ADCSRA |= (1 << ADSC); // ADC çevirme işlemini başlat (ADSC=1).
        while(ADCSRA & (1 << ADSC));
                                         // Okuma işlemi bitene kadar bekle (ADSC=0).
        cvalue = ADC; // ADC değerini oku.
        tempC = (cvalue - 347) / 0.7 ; // Sıcaklık değerine dönüştür.
        itoa (cvalue, showcvalue, 10); // cvalue sayısını karakter dizisi yap.
        sprintf (showtvalue,"%3.0f %cC",tempC, (char)223);
                                         // tempC sıcaklığını karakter dizisi yap.
        LCD_Clear(); // LCD'yi temizle.
        LCD_Print("ADC : "); // "ADC: " yaz.
```

```

LCD_Print(showcvalue);           // ADC değerini yaz.
LCD_Action(0xC0);                 // İkinci satır, birinci sütuna git.
LCD_Print("Sic.: ");             // "Sic.: " yaz.
LCD_Print(showtvalue);           // Sıcaklık değerini yaz.
_delay_ms(1000);                 // 1 000 ms bekle.
    }
}

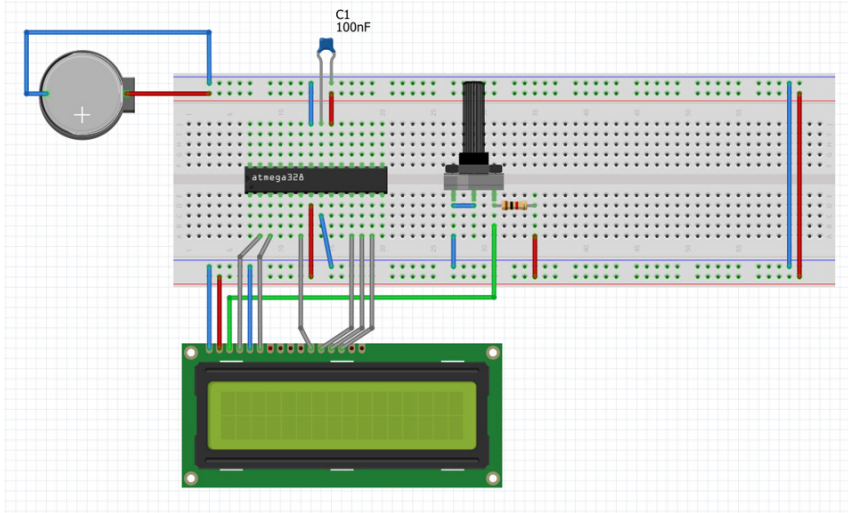
```

5. Adım: Kodlar içerisinde kullandığımız *sprintf()* fonksiyonunun doğru çalışabilmesi için 4.2 No.lu uygulamanın 5 ve 6. adımlarında verilen ayarlamaları yapınız.

6. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyası proje klasörünüzde **Debug** dizini içerisinde bulunur.

7. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

8. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.69'da verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.69: Uygulama 4.10 için breadboard üzerindeki devre

9. Adım: Soğuk ve sıcak ortamda devrenin çalışmasını izleyiniz. Güvenilir sonuçlar veren bir termometre ile ortam sıcaklığını ölçünüz. Fark varsa sıcaklık hesaplama işleminin gerçekleştirildiği aşağıdaki satırda "0.7" değerini artırarak / azaltarak devreyi kalibre ediniz.

```
tempC = (cvalue - 347) / 0.7 ;           // Sıcaklık değerine dönüştür.
```

10. Adım: Güç kaynağını kapatınız.

11. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
3. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
4. Ortam sıcaklığı değiştirilerek LCD üzerindeki değerlerin değişimi tespit edildi.		
5. Ortam sıcaklığı bir termometre yardımı ile ölçülerek sıcaklık değerini hesaplayan formül yeniden düzenlendi.		
6. Devrenin kalibrasyonu sağlandı.		
7. Temizliğe dikkat edildi.		



SIRA SİZDE

4.10 No.lu uygulama için verilen devrede, PORTB0 ve PORTB1 pinlerine iki adet LED bağlayınız. Sıcaklık değeri 25 °C'nin altında iken PORTB0 pinine bağlı birinci LED'in, 25 °C ve üzerinde ise PORTB1 pinine bağlı ikinci LED'in yanmasını sağlayan gömülü kodu yazınız ve devreyi kurarak çalıştırınız.

4.2.2. Haricî Sıcaklık Algılayıcısı

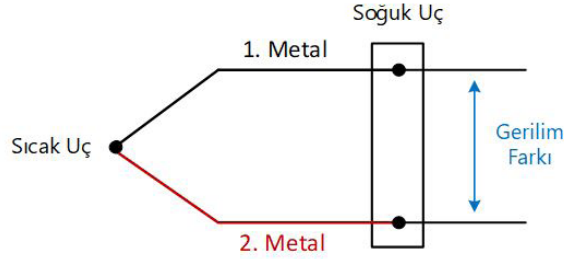
Haricî sıcaklık algılayıcılar, endüstriyel süreç denetiminde ve birçok elektronik cihazda yaygın bir şekilde kullanılmaktadır. Farklı amaçlara yönelik, farklı türlerde sıcaklık algılayıcıları mevcuttur. Bir haricî sıcaklık algılayıcısı kullanılarak mikrodenetleyici yardımıyla sıcaklık ölçümü yapılabilir.

4.2.2.1. Haricî Sıcaklık Algılayıcı Türleri

Sıcaklık algılayıcıları, ölçüm yapılacak ortam ve amaca göre farklı türlerde üretilmektedir. Sıcaklık algılayıcılarını aşağıdaki türlerde gruplandırmak mümkündür:

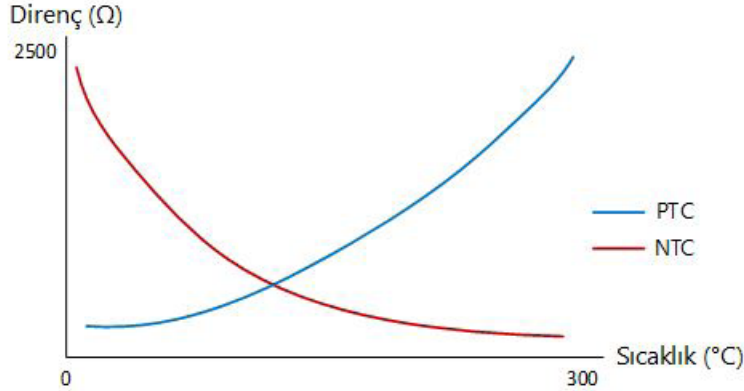
1. Isıl Çift (Termokupl): İki farklı türde metalin bir noktada birleştirilmesi ile oluşturulmuştur. Birleşim noktasının olduğu uç, **sıcak uç**; diğeri ise **soğuk uç** olarak adlandırılır. Sıcak uç ısıtıldığında metal içerisindeki elektronların hareketi sonucunda uçlarda, çok düşük seviyede bir gerilim farkı

meydana gelir. Çoğunlukla yüksek ısı değişimlerinin ölçüldüğü endüstriyel süreçlerde, ısı çifti tercih edilir. Isıl çiftler, üzerindeki metal türüne göre -200 °C ile 2 000 °C arasında ölçüm yapabilir. Isıl çiftin yapısı Görsel 4.70’te verilmiştir.



Görsel 4.70: Isıl çiftin yapısı

2. Termistör: Sıcaklık ile direnci değişen yarıiletken bir malzemedir. **Pozitif direnç katsayılı (PTC)** ve **negatif direnç katsayılı (NTC)** olmak üzere iki türü bulunur. PTC’nin sıcaklık ile direnci artar, NTC’nin sıcaklık ile direnci azalır. Direncin değeri ortam sıcaklığını verir. Bu nedenle termistörün üzerinden akım geçtiğinde gerilim değeri ölçülerek sıcaklık düzeyi tespit edilir. Termistörün çıkış karakteristiği doğrusal değildir. Sıcaklığa duyarlı devrelerde daha çok, akım sınırlayıcı olarak kullanılır. PTC ve NTC türleri için sıcaklığa göre direncin değişimi temsili olarak Görsel 4.71’de verilmiştir.



Görsel 4.71: PTC ve NTC’nin çalışma karakteristikleri

3. Direnç Sıcaklık Detektörü (RTD): İletken malzemelerin bir sargı şeklinde sarılması ile gerçekleştirilen sıcaklık algılayıcı türüdür. Pozitif katsayılı direnç (PTC) türünden olduğu için sıcaklık arttıkça direnç değeri artar. Termistöre göre daha yüksek sıcaklıkları (-200-600 °C) ölçebilir ancak ısı çifti göre sıcaklık aralığı daha düşüktür. Maliyetleri ısı çifti göre daha yüksek olup **PT100** olarak da adlandırılır. İki, üç ve dört telli olan türleri bulunmakta olup tel sayısına göre ölçüm hassasiyeti artar. Doğrusal bir karakteristiğe sahip olduğundan birçok uygulamada sıcaklık ölçümü için tercih edilir.

4. Yarıiletken Sıcaklık Algılayıcı: Yarıiletken malzemelerin kullanıldığı hassasiyeti yüksek, sıcaklık algılayıcı türüdür. Bu tür sıcaklık algılayıcılar, sıcaklık ölçümü gereken birçok elektronik sistemde tercih edilir ancak yüksek sıcaklık aralığında ölçüm yapamayıp -55 °C ile +150 °C arasında çalışabilir. Örneğin LM35, mikrodenetleyici uygulamalarında sık kullanılan yarıiletken sıcaklık algılayıcılardan biridir.

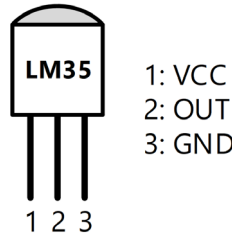


SIRA SİZDE

Bir üretim tesisinde gerçekleştirilen süreç kontrolünde, 25 °C ile 300 °C arasındaki sıcaklık değerini mümkün olduğunca hassas ve doğrusal bir şekilde ölçmek amacıyla hangi sıcaklık algılayıcı türü seçilebilir? Nedenini belirtiniz.

4.2.2.2. Yarıiletken Sıcaklık Algılayıcıların Kullanımı

Sıcaklık ölçümünün gerekli olduğu birçok elektronik uygulamasında yaygın olarak kullanılan LM35, hassasiyeti yüksek ve gerilim çıkışlı bir sıcaklık algılayıcısıdır. Oda sıcaklığında ± 0.5 °C doğrulukla -55 °C ile +150 °C arasında ölçüm yapabilir. Çıkış, 1 °C'de bir doğrusal olarak 10 mV değişim gösterir. 4 V ile 30 V arasında gerilim ile çalışabilir. +5 V besleme gerilimi uygulandığında, çıkış değeri doğrudan mikrodenetleyici ADC modülü ile ölçülebilir. LM35 bacak bağlantıları, Görsel 4.72'de verilmiştir.



Görsel 4.72: LM35 bacak bağlantısı



UYGULAMA

Adı:	Haricî Sıcaklık Algılayıcı ile Sıcaklık Ölçümü	No : 4.11
AMAÇ:	Uygun sıcaklık algılayıcısı ile program yazdırmak ve yazılan programı test ettirmek.	Süre: 60 dk.

Görev: İş sağlığı ve güvenliği kurallarına dikkat ederek aşağıda verilen adımlar doğrultusunda, mikrodenetleyici ADC modülü üzerinden, uygun bir sıcaklık algılayıcı ile sıcaklık ölçümü gerçekleştiriniz.

Kullanılacak Araç Gereç: Tablo 4.41’de belirtilmiştir.

Tablo 4.41: Uygulama 4.11 İçin Malzeme Listesi

Adı	Özelliği	Miktarı
Mikrodenetleyici entegresi	Atmega328, 28 pinli DIP soket	1 adet
Sıcaklık algılayıcı	LM35	1 adet
LCD	16x2 karakter	1 adet
Direnç	1 k Ω	1 adet
Kondansatör	100 nF	1 adet
Potansiyometre	5 k Ω	1 adet
Breadboard		1 adet
DC güç kaynağı	5 V	1 adet
Mikrodenetleyici programlayıcı	Atmega328P mikrodenetleyici için	1 adet
Bağlantı kablosu		1 m
Yan keski		1 adet

Uygulama Adımları:

- 1. Adım:** 4.1 No.lu uygulamanın 1. adımında verilen şekilde, **Atmel Studio** programında **ADC_Uyg_02** adlı yeni bir proje oluşturunuz.
- 2. Adım:** 4.1 No.lu uygulamanın 2. adımında verilen şekilde, **Device Selection** penceresinden **Atmega328P** mikrodenetleyicisini seçiniz.
- 3. Adım:** Projenizde **main.c** dosyasının bulunduğu dizinde **Include** adında bir dizin oluşturup içerisine 4.2 No.lu uygulamanın 3. adımında verilen **lcd.h** adındaki dosyayı ekleyiniz. Bu başlık dosyasının LCD’yi kullanabilmeniz için gerekli olduğunu hatırlayınız.
- 4. Adım:** **main.c** dosyası içerisine aşağıda verilen program kodlarını yazınız.

```

#include <avr/io.h>
#include <stdio.h>
#include <stdlib.h>
#include ".\Include\lcd.h"           // lcd.h dosyasını dahil et.

int main(void)
{
    volatile int cvalue;             // ADC çevirme sonucunun kaydedildiği değişken
    volatile float tempC;           // Gerilim değerinin kaydedildiği değişken
    char showcvalue [16];           // ADC çevirme sonucu karakter dizisi
    char showtvalue [16];           // Gerilim değeri karakter dizisi
    LCD_Init();                     // LCD'yi tanı.
    ADMUX &= ~(1 << REFS1); // Referans gerilimi seçimi: REFS[1:0]=01:
    ADMUX |= (1 << REFS0); // AREF pininde haricî kondansatörlü AVCC
    ADMUX &= ~(1 << ADLAR); // Sonuç hizalama (ADLAR=0: sağa yaslı)
    ADMUX &= ~((1 << MUX3) | (1 << MUX2) | (1 << MUX1) | (1 << MUX0)); // ADC giriş kanalı seçimi (MUX[3:0]=0000: ADC0)
    ADCSRA |= (1 << ADEN); // ADC etkin (ADEN=1).
    ADCSRA &= ~(1 << ADATE); // Otomatik tetikleme devre dışı (ADATE=0).
    ADCSRA &= ~((1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0)); // Prescaler (Ön ölçekleyici) (ADPS[2:0]=000: /2)

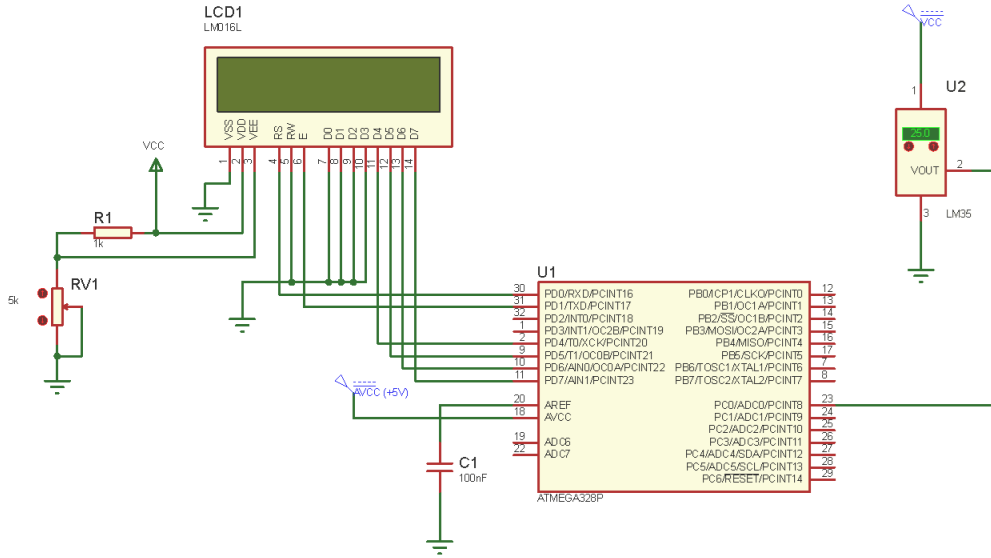
    while (1)
    {
        ADCSRA |= (1 << ADSC); // ADC çevirme işlemini başlat (ADSC=1)
        while(ADCSRA & (1 << ADSC)); // Okuma işlemi bitene kadar bekle (ADSC=0).
        cvalue = ADC; // ADC değerini oku.
        tempC = cvalue * 0.4875; // Sıcaklık değerini hesapla.
        itoa (cvalue, showcvalue,10); // cvalue sayısını karakter dizisi yap.
        sprintf (showtvalue,"%2.1f %cC", tempC, (char)223); // tempC sıcaklığını karakter dizisi yap.
        LCD_Clear(); // LCD'yi temizle.
        LCD_Print("ADC : "); // "ADC: " yaz.
        LCD_Print(showcvalue); // ADC değerini yaz.
        LCD_Action(0xC0); // İkinci satır, birinci sütuna git.
        LCD_Print("Sic.: "); // "Sic.:" yaz.
        LCD_Print(showtvalue); // Sıcaklık değerini yaz
        _delay_ms(1000); // 1 000 ms bekle
    }
    return 0;
}

```

5. Adım: Kodlar içerisinde kullandığımız *sprintf()* fonksiyonunun doğru çalışabilmesi için 4.2 No.lu uygulamanın 5 ve 6. adımlarında verilen ayarlamaları yapınız.

6. Adım: Yazdığınız programı, **Build** menüsünden **Build Solution** seçeneği ile derleyiniz. HEX dosyasının proje klasörünüzde **Debug** dizini içerisinde bulunduğunu hatırlayınız.

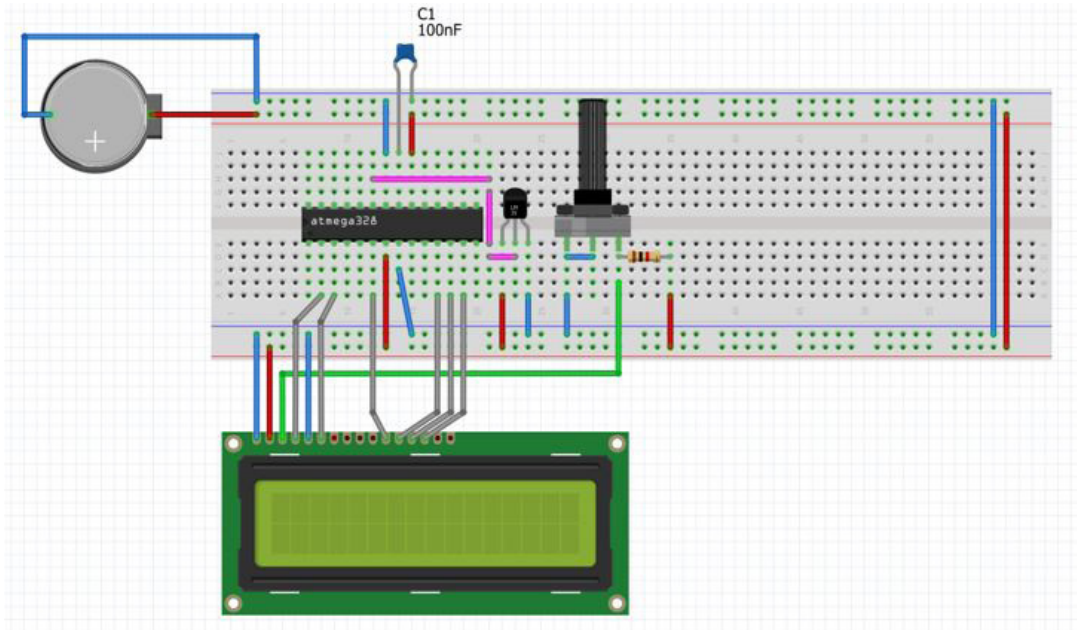
7. Adım: Devre simülasyon programında, Görsel 4.73'te verilen devreyi kurarak simüle ediniz.



Görsel 4.73: Uygulama 4.11 için simülasyon devre şeması

8. Adım: İş güvenliği tedbirlerini alarak mikrodenetleyiciyi programlayınız.

9. Adım: İş güvenliği tedbirlerini alarak devreyi breadboard üzerine Görsel 4.74'te verildiği gibi kurunuz ve öğretmeninizden onay aldıktan sonra çalıştırınız.



Görsel 4.74: Uygulama 4.5 için breadboard üzerindeki devre

10. Adım: Sıcaklık algılayıcısının sıcaklığını değiştirerek LCD’de yazan değerleri izleyiniz. Soğuk ve sıcak ortamda devrenin çalışmasını izleyiniz. Güvenilir sonuçlar veren bir termometre ile ortam sıcaklığını ölçünüz. Fark varsa sıcaklık hesaplama işleminin gerçekleştirildiği aşağıdaki satırda “0.4875” değerini değiştirerek devreyi kalibre ediniz.

tempC = cvalue * 0.4875; // Sıcaklık değerini hesapla.

11. Adım: Güç kaynağını kapatınız.

12. Adım: İş güvenliği tedbirlerini alarak çalışma alanınızı temizleyiniz.

DEĞERLENDİRME

Çalışmalarınız kontrol listesi kullanılarak değerlendirilecektir. Çalışmalarınızı yaparken değerlendirme ölçütlerini dikkate alınız.

KONTROL LİSTESİ

Ölçütler	Evet	Hayır
1. Çalışmada iş sağlığı ve güvenliği kurallarına dikkat edildi.		
2. Devre simülasyonunda LM35’in sıcaklık değeri değiştirilerek LCD üzerindeki değerlerin değişimi tespit edildi.		
3. Mikrodenetleyici yazılımı doğru bir şekilde kodlanarak derlendi.		
4. Devre, breadboard üzerine doğru bir şekilde kuruldu.		
5. Deneyde ortam sıcaklığı değiştirilerek LCD üzerindeki değerlerin değişimi izlendi.		
6. Ortam sıcaklığı bir termometre yardımı ile ölçülerek sıcaklık değerini hesaplayan formül yeniden düzenlendi ve devrenin kalibrasyonu sağlandı.		
7. Temizliğe dikkat edildi.		



SIRA SİZDE

4.11 No.lu uygulamada verilen devreye bir fan ekleyiniz. Eklediğiniz fanın sıcaklık arttıkça daha hızlı dönmesini, sıcaklık azaldıkça yavaşlamasını sağlamak üzere devreyi yeniden tasarlayarak gömülü yazılımını kodlayınız. Fan için minyatür DC motor kullanınız. Tasarladığınız devreyi kurarak deneyiniz.



ÖLÇME VE DEĞERLENDİRME 4

A) Aşağıda verilen cümlelerin başındaki boşluğa, cümlede yer alan ifade doğru ise “D”, yanlış ise “Y” yazınız.

1. () Analog işaretler, yalnızca 0 ve 1 değerlerini alır.
2. () Örnekleme frekansı (f_s), örnekleme periyodunun (T_s) çarpmaya göre tersi alınarak bulunur.
3. () Kuantalama işlemi sonucunda, ölçülen işaret belirli seviyelerde değerlendirilir.
4. () 10 bitlik bir ADC'nin üretebileceği en yüksek sayısal çıkış kodu değeri 1023'tür.
5. () Kesme rutini icra edildikten sonra, program en baştan başlar.
6. () Z/S çalışırken kesme kullanılırsa başka komutlar icra edilir.

B) Aşağıdaki cümlelerde bulunan boşlukları uygun kelimelerle doldurunuz.

7. Bir algılayıcıdan 1 saniyede 50 örnek alınıyorsa örnekleme frekansı Hz'dir.
8. Bir işaret, her 50 ms'de bir tanımlı değere sahipse bu işarete denir.
9. İlk ADC çevrim işlemi, kurulum işlemleri nedeniyle saat darbesinde yapılır.
10. ADC çevirme işlemini başlatmak için adlı kaydedici kullanılır.
11. Bir olayın gerçekleştiğini *while(1) {}* sürekli döngüsü olmadan otomatik olarak takip etmek için tanımlanır.
12. Mikrodenetleyici port pinlerinden çekilebilecek en yüksek kaynak (source) akımı, yaklaşık mA'dir.
13. Z/S1 modülünde Z/S değeri adlı kaydedicide tutulur.
14. TCNTn ve değerlerinin birbirine eşit olması durumunda karşılaştırma eşleşmesi oluşur.

C) Aşağıda verilen kesme bayrakları ve anlamlarını, kesme bayraklarının başındaki boşluğa uygun olan harfleri yazarak eşleştiriniz.

15.

Kesme Bayrakları	Anlamları
(...) I. ICFn	a) Z/S çıkış karşılaştırma kesmesi
(...) II. OCFnx	b) Z/S taşma kesmesi
(...) III. TOVn	c) Analog/sayısal çevirme kesmesi
(...) IV. ADIF	d) Z/S giriş yakalama kesmesi

D) Aşağıdaki soruları dikkatlice okuyarak doğru seçeneği işaretleyiniz (Kaydedici adlarında geçen **n** indisi zamanlayıcı / sayıcının numarası, **x** ise kanal adıdır.).

16. Z/S değerinin bir değerle karşılaştırılması ve isteniyorsa çıkış dalga şeklinin değiştirilmesi işlemi aşağıdakilerden hangisidir?

- A) ADC çevirme
- B) Asenkron çalışma
- C) Çıkış karşılaştırma
- D) Giriş yakalama
- E) Ön ölçekleme

17. Aşağıdaki Z/S çalışma modlarından hangisinde sayıcı ileriye doğru sayarken TCNTn değeri OCRnx kaydedicisindeki değere eşit olduğunda sayıcı değeri sıfırlanır?

- A) CTC modu
- B) Doğru faz PWM modu
- C) Doğru frekans ve faz PWM modu
- D) Hızlı PWM modu
- E) Normal mod

18. Aşağıdakilerden hangisi bir Z/S kesmesi değildir?

- A) Çıkış karşılaştırma A
- B) Çıkış karşılaştırma B
- C) Giriş yakalama
- D) Haricî kesme
- E) Taşma

19. Z/S kesmeleri aşağıdaki kaydedicilerden hangisi üzerinden etkinleştirilir?

- A) OCRnx
B) TCCRnB
C) TCNTn
D) TIFRn
E) TIMSKn

20. İşaretin üretildiği zaman diliminde sürekli bir aralıkta birbirinden farklı birçok değer alabilen işaret aşağıdakilerden hangisidir?

- A) Ayrık zamanlı işaret
B) Kuantalama işareti
C) PWM işareti
D) Sayısal işaret
E) Sürekli zamanlı işaret

E) Aşağıda verilen soruları cevaplayınız.

21. 12 bitlik bir ADC'nin üretebileceği sayısal çıkış kod aralığı nedir?

22. Referans gerilimi 5 V olan 8 bitlik bir ADC'nin çıkış kodu 11011001 ise yaklaşık giriş gerilimi ne kadardır? Hesaplayınız.

23. Çalışma saat darbesi frekansı $f_{IO} = 1$ MHz olan bir mikrodenetleyicide bulunan ADC için ön ölçekleme bölme faktörü 8 olarak seçilmiştir. Örnekleme frekansı (f_s) ve örnekleme periyodunu (T_s) hesaplayınız.

24. Bir ışık algılayıcısından örnekleme frekansı 100 kHz olan 8 bitlik bir ADC yardımıyla örnek alınmaktadır. Ölçülen değerler için mümkün olan en küçük ve en büyük değer aralığı nedir? Örnekleme periyodunu (T_s) hesaplayınız.

25. Bir motorun hız değerleri belirli aralıklar ile örneklenmektedir. Motorun hızı 0 ile 2048 RPM arasında değiştiğine göre, 10 bitlik bir ADC kullanılması hâlinde, sayısal çıkış kodundaki her bir bit değişimi kaç RPM değerine karşılık gelir?

26. 8 bitlik bir Z/S (örneğin Z/S0) doğru faz (phase correct) PWM modunda çalıştırılıyor. $f_{CLK_IO} = 4$ MHz ve ön ölçekleyici değeri $n = 8$ iken PWM çıkış frekansını hesaplayınız.

27. $f_{CLK_IO} = 8$ MHz ve ön ölçekleyici değeri $n = 64$ iken, 16 bitlik bir Z/S değerinin en yüksek değere ulaşması için gerekli süreyi (periyodu) hesaplayınız.

KAYNAKÇA

- AEQ Web (2021), Atmega328 16x2 4-Bit LCD Display, Wwww.Aeq-Web.Com.
- Altınbaşak O. (2017), Mikrodenetleyiciler ve PIC Programlama
- A. S. Sedra, K. C. Smith (2011), Microelectronic Circuits, 6th Edition, Oxford University Press, Pp. 4-13.
- B. Cincorap (2016). Atmel Programlama 7- Usbasp Kurulumu. <https://bariscincorop.blogspot.com/2016/01/Atmel-Avr-7-Usbasp-Kurulumu.html>
- Bereket Metin, Tekin Engin (2015). Dijital Elektronik
- Bilişim Teknolojileri Alanı Çerçeve Öğretim Programı (2017). Ankara: Millî Eğitim Bakanlığı Yayınları.
- Bilişim Teknolojileri Alanı Ders Bilgi Formu (2020). Ankara: Millî Eğitim Bakanlığı Yayınları.
- D. Hüseyin (2012). Dijital Elektronik
- D. Zhu, T. Sifleet, T. Nunnally, Y. Huang (2021), Analog To Digital Converters, Gatech University, https://Ume.Gatech.Edu/Mechatronics_Course/ADC_F08.Pdf.
- G. Dökmetaş (2018). C ile AVR Programlama. <http://www.lojikprob.com/avr/c-ile-avr-programlama-60-butun-derslerin-listesi/>
- Microchip Technology (2020), Atmega48a/PA/88A/PA/168A/PA/328/P Megaavr Data Sheet (DS40002061B), Pp. 1-653.
- National Semiconductor (1999), DAC0800/DAC0802 8-Bit Digital-To-Analog Converters Data Sheet (DS005686), Pp. 1-11.
- R. Çevik (2021), Sıcaklık Sensörleri Nedir? Çeşitleri Nelerdir?, <https://www.elektrikport.com/makale-detay/sicaklik-sensorleri-nedir-cesitleri-Nelerdir/22055>.
- S. Çiçek (2012). CCS C ile PIC Programlama
- Texas Instruments (2017), LM35 Precision Centigrade Temperature Sensors Data Sheet (SNIS159H), Pp. 1-38.
- M. Yağımlı, F. Akar(2014). Dijital Elektronik

GENEL AĞ KAYNAKÇASI VE GÖRSEL KAYNAKÇA

<http://kitap.eba.gov.tr/karekod/Kaynak.php?KOD=2417>



ÖĞRENME BİRİMLERİ ÖLÇME VE DEĞERLENDİRME CEVAP ANAHTARLARI

1. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

- A)** 1. D 2. Y 3. D 4. D 5. Y
- B)** 6. Onaltılık 7. LSB 8. VEYA 9. 1 (Bir) 10. 5 volt
- C)** 11. E 12. B 13. A 14. D

2. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

- A)** 1. Y 2. D 3. D 4. D 5. Y 6. D 7. Y 8. Y 9. D 10. D
- B)** 11. Harvard 12. Kristal 13. PINx 14. Power-On Reset 15. C 16. PWM 17. PORTx
18. delay.h 19. Statik RAM'dir. 20. CKOUT
- C)** 21. C 22. B 23. C 24. A 25. B 26. B 27. B 28. D 29. A 30. D

3. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

- A)** 1. Y 2. D 3. Y 4. D 5. D 6. Y 7. D
- B)** 8. açılış gecikmesi 9. kontak 10. SSR 11. faz koruma 12. DC motorlar
13. Step motorlar 14. baud rate 15. 14 16. I²C
- C)** 17. E 18. D 19. E 20. C 21. E 22. A 23. C 24. B 25. A

4. ÖĞRENME BİRİMİNİN CEVAP ANAHTARI

- A)** 1. Y 2. D 3. D 4. D 5. Y 6. D
- B)** 7. 50 8. ayrık zamanlı işaret 9. 25 10. ADCSRA 11. kesme 12. 20 13. TCNT1
14. OCRnx
- C)** 15. I- d II- a III- b IV- a
- D)** 16. C 17. A 18. D 19. E 20. E
- E)** 21. 0-4095 22. ~4.25 V 23. 125 kHz – 8 µs 24. 0-255 – Ts=10 µs 25. 2 RPM
26. 980.39 kHz 27. ~0.52 µs