KOCAELİ ÜNİVERSİTESİ
MEKATRONİK MÜHENDİSLİĞİ
ENDÜSTRİYEL OTOMASYON HABERLEŞMESİ DERSİ
PROJE RAPORU

PIC 16F877 İle RS-485 Seri Haberleşme Uygulaması

Maksut KAYA - 090224040
Heval Yusuf DELİL - 090224023
Ahmet YILDIZ-100224002

26.12.2014

Yrd.Doç.Dr. Selçuk KİZİR

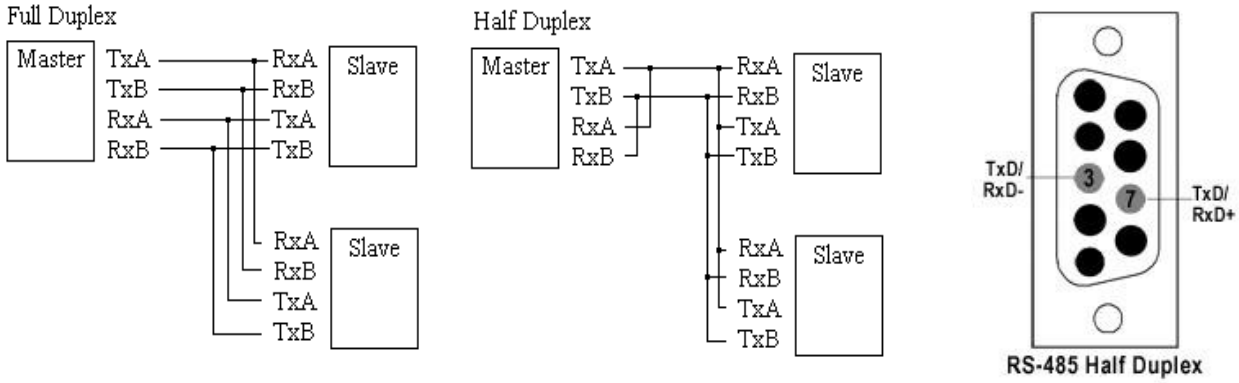
İçindekiler

RS-485 Hakkında Kısa Bilgi

1200 m'ye kadar kablo uzunluğuna izin veren, çok noktalı, half-dublex, seri iletişim veri yolu standardır.

RS-232 standardının uzun mesafelisi olarak düşünülebilir. RS-232, 5 metreye kadar kablo uzunluklarının desteklerken, RS-485'te bu uzunluk çok daha fazladır. İletişim hızı kullanılacak kablo uzunluğu ve türüne göre değişkendir. Bağlantılarda, üreticiye ve adres yolu türüne bağlı olarak çeşitli kablolar kullanılabilir.

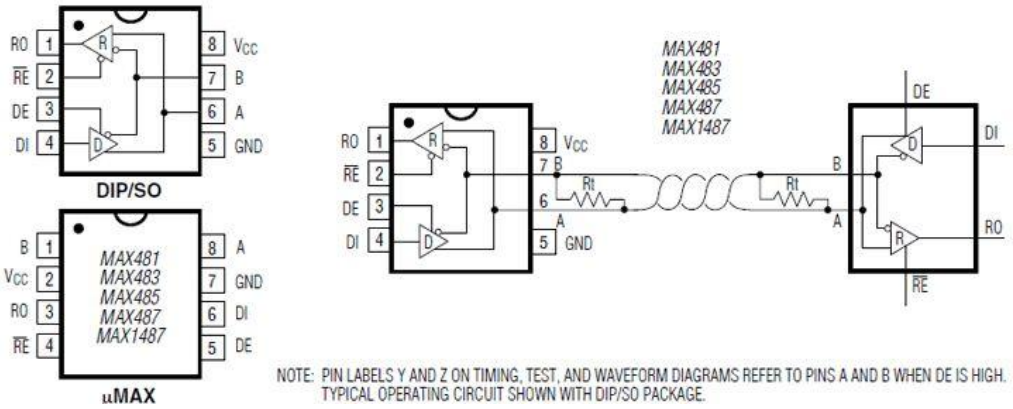
RS485 in network yapısı data işleme ve kontrol uygulamalarında yoğun bir şekilde kullanılmasının ana nedenidir. 12 kohm giriş direnci ile networke 32 cihaza kadar bağlantı yapılabilir. Daha yüksek giriş direnciyle bu sayı 256 ya kadar çıkarılabilir. RS485 tekrarlayıcıları ile bağlanabilecek cihaz sayısı birkaç bine, haberleşme mesafeside birkaç kilometreye çıkabilir. RS485 bunun için ayrıca bir donanım istemez yazılım kısmıda RS232 den zor değildir.



RS485 in belli başlı teknik özellikleri

- Maksimum sürücü sayısı : 32
- Maksimum alıcı sayısı : 32
- Çalışma şekli : Half Duplex
- Network Yapısı : Çok noktalı bağlantı
- Maksimum Çalışma Mesafesi : 1200 metre
- 15-12 m kablo uzunluğunda maksimum hız : 35 Mbps
- 1200 m kablo uzunluğunda maksimum hız : 100 kbps
- Alıcı giriş direnci : 12 kohm
- Alıcı giriş duyarlılığı :+/-200 mvolt
- Alıcı giriş aralığı : -7…12 volt
- Maksimum sürücü çıkış voltajı : -7…12 volt
- Minimum sürücü çıkış voltajı (yük bağlı durumda) : +/-1.5 volt

RS-485 Half-Dublex



NOTE: PIN LABELS Y AND Z ON TIMING, TEST, AND WAVEFORM DIAGRAMS REFER TO PINS A AND B WHEN DE IS HIGH. TYPICAL OPERATING CIRCUIT SHOWN WITH DIP/SO PACKAGE.

Hata Kontrolü denetimi

Checksum

- Veriyi oluşturan karakterler, kullanıcının isteğine bağlı olarak gruplandırılır.
- Bu gruplar 16'lık tabandaki sayılar olarak ele alınır.
- Bu sayıların toplamı bulunur.
- Bu toplam verinin sonuna eklenir.

H	e	l	l	o		w	o	r	l	d	.
48	65	6C	6C	6F	20	77	6F	72	6C	64	2E

4865 + 6C6C + 6F20 + 776F + 726C + 642E + carry = 71FC

- Veri gönderilir.
- Veriyi alan taraf, aynı veri üzerinde aynı işlemleri yapar.
- Elde ettiği sonucu gelen değer ile karşılaştırır.
- İki değer aynı ise, veri hatasız iletilmiştir.

H	e	l	l	o		w	o	r	l	d	.
48	65	6C	6C	6F	20	77	6F	72	6C	64	2E

1. PIC16F877A İle RS-485 Uygulaması

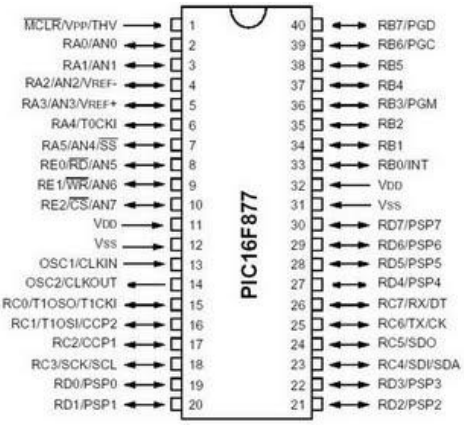
Projenin tanımı

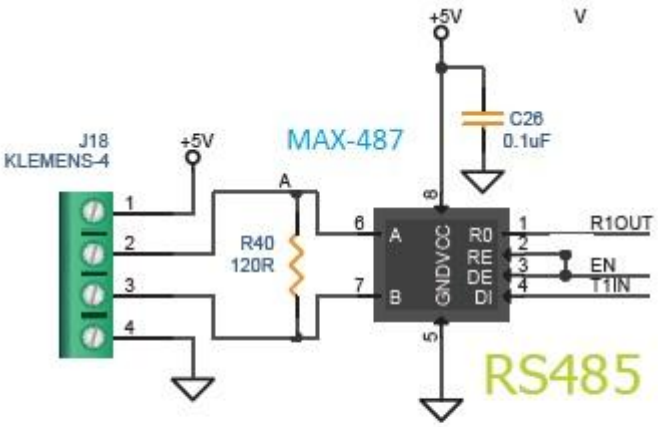
Proje kapsamında bir master ve üç slaveden oluşan sistemin, ortam sıcaklıklarını ölçerek mastera göndermesi ve masterdan bu sıcaklıkların kontrolü RS-485 kullanılarak yapılmaktadır.

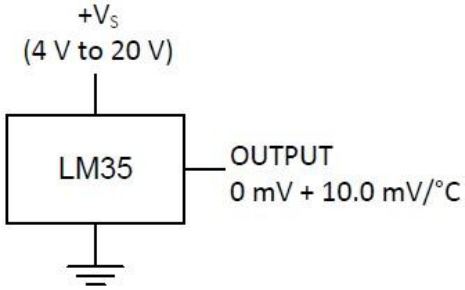
- ✓ Proje devresindeki slaveler 3 ayrı odayı temsil etmektedir.
- ✓ Slave olarak kullanılan PIC16F877 mikrodenetleyiclerine LM35 sıcaklık sensörü bağlanarak ortamların sıcaklıkları belirlendi.
- ✓ Ölçülen sıcaklık bilgisi RS-485 ile mastera gönderildi.
- ✓ Mastera gönderilen sıcaklı değerleri grafik LCD de üç ayrı odayı temsil edecek şekilde gösterildi.
- ✓ Slaveden gönderilen sıcaklık değeri istenilen değerden fazla ise soğutucu değilse ısıtıcı aktif edildi.
- ✓ Gönderilen sıcaklık değerleri istenilen sıcaklıktaysa devreye konulan ısıtıcı ve soğutucu fan girişleri aktif edilmedi.

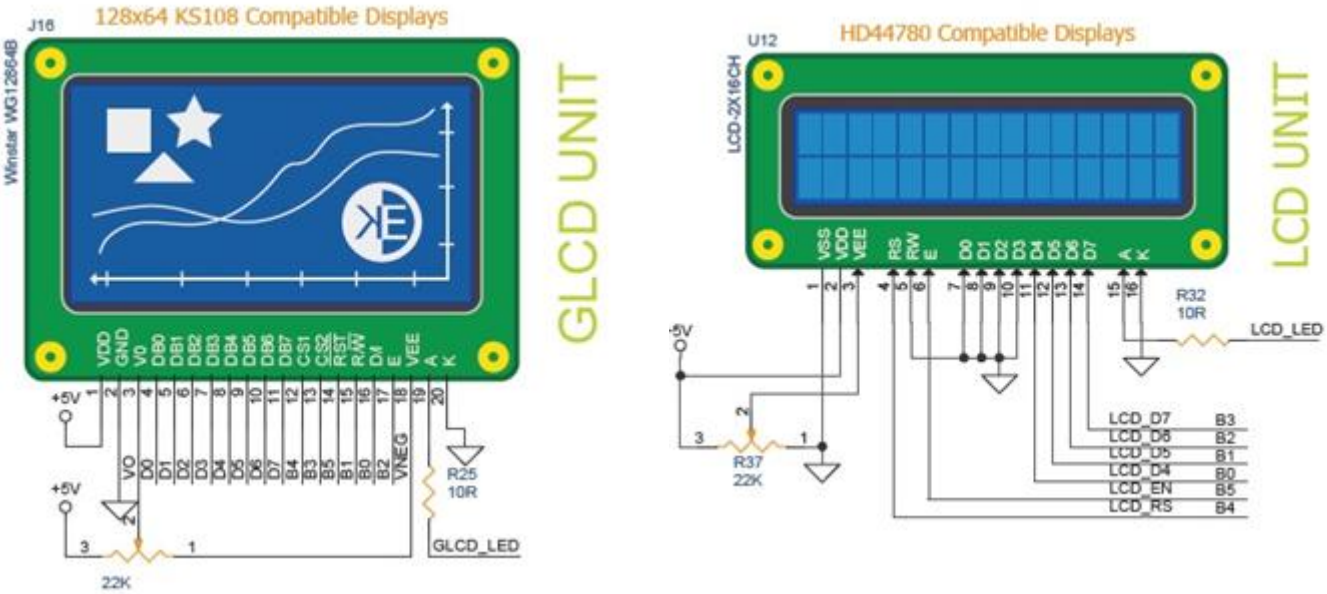
2.2.Projede kullanılan malzemeler

- 4x PIC 16F877A
- 4x MAX487
- 3x LM35
- 3x 16*2 LCD
- 1x 128*64 GLCD

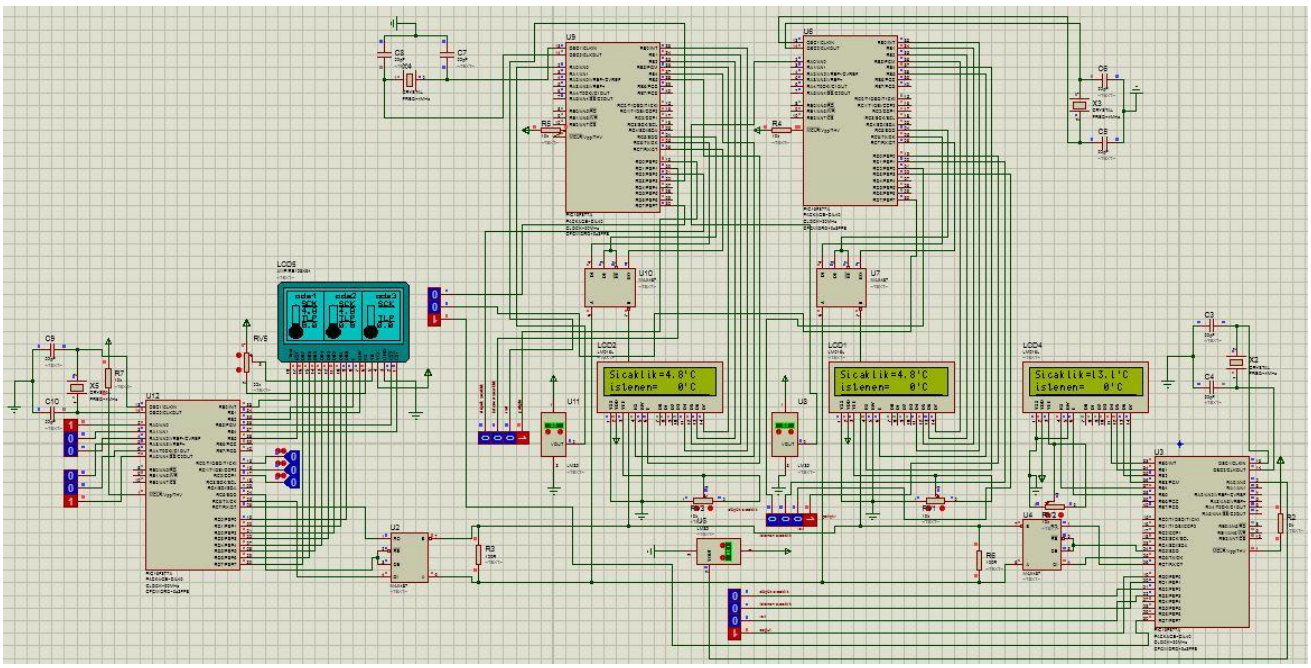








Proteus Görüntüsü



Yazılım

LCD kütüphanesi

```
// bu LCD sürücü dosyası 2x16 HD44780 uyumlu
EXM1 kiti için yazılmıştır.
// RB0 --> LCD'nin D4 ucuna
// RB1 --> LCD'nin D5 ucuna
// RB2 --> LCD'nin D6 ucuna
// RB3 --> LCD'nin D7 ucuna
// RB4 --> LCD'nin RS ucuna
// RB5 --> LCD'nin E ucuna
// R/W  ucu direkt şaseye bağlanacaktır.

#define e pin_b5
#define rs pin_b4
void lcd_komut (byte komut)
{
  output_b(komut>>4);
  output_low(rs);
  delay_cycles(1);
  output_high(e);
  delay_cycles(1);
  output_low(e);
  delay_ms(2);

  output_b(komut&0x0F);
  output_low(rs);
  delay_cycles(1);
  output_high(e);
  delay_cycles(1);
  output_low(e);
  delay_ms(2);

}

void lcd_veri (byte veri)
{

  output_b(veri>>4);
  output_high(rs);
  delay_cycles(1);
  output_high(e);
  delay_cycles(1);
  output_low(e);
  delay_ms(2);

  output_b(veri&0x0F);
  output_high(rs);
  delay_cycles(1);
  output_high(e);
  delay_cycles(1);
  output_low(e);
  delay_ms(2);

}
```

```
void imlec (byte satir, byte sutun)
{
  if (satir == 1)
     lcd_komut(0x80 | (sutun-1));
  if (satir == 2)
     lcd_komut(0x80 | (0x40 + (sutun-1)));
}

void lcd_hazirla ()
{
  int i=0;
  output_low(rs);
  output_low(e);
  delay_ms(30);

  for(i=0;i<=3;i++)
  {
     lcd_komut(0x03);
     delay_ms(5);
  }

  lcd_komut(0x02);
  lcd_komut(0x28);
  lcd_komut(0x08);
  lcd_komut(0x0C);
  lcd_komut(0x06);
  lcd_komut(0x01);
}
```

Master

```
#include <16f877.h>     // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.


#fuses HS,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD

#use delay (clock=20000000) // Gecikme fonksiyonu için kullanılacak osilatör frekansı belirtiliyor.

#use rs232 (baud=9600, xmit=pin_C6, rcv=pin_C7, parity=N, stop=1, TIMEOUT=25) // RS232 protokolünün
9600 bit/sn baud hızında olacağını ve

                                                // TX,RX uçlarının hangi pinler olacağını tanımlıyor
#define gon pin_C5
#include <kit_glcd.c> //
#include <graphics.c>  // graphics.c dosyası programa ekleniyor

#define DGS pin_C0


#priority int_ccp1,int_ccp2,int_rda

int   i=0;
float n1,n2,n3;
char t1,t2,t3,t4,t5,sd[3];
char yazi1[]="oda1";
char yazi2[]="oda2";
char yazi3[]="oda3";
char tlp1[]="TLP";
char sck1[]="SCK";

char isi1[5];
char isi2[5];
char isi3[5];


char ist1[5];
char ist2[5];
char ist3[5];

int bar1,bar2,bar3;
char ccsum1[],ccsum2[],ccsum3[];
char ccsum_s1[],ccsum_s2[],ccsum_s3[];
char ccsum_ns1[],ccsum_ns2[],ccsum_ns3[];


#INT_CCP1
void  CCP1_isr(void)
```

```
{
    if(sd[i]==25) sd[i]=24;
    sd[i]++;
    write_eeprom((i),(sd[i]));
}

#INT_CCP2
void  CCP2_isr(void)
{
//!set_timer1(0);
if(sd[i]==0) sd[i]=1;
sd[i]--;
write_eeprom((i),sd[i]);
}

#int_RDA
void seri()
{

  t1 = getc();
  if(t1=='X')   output_high(pin_a3);
  else if(t1=='T')   output_low(pin_a3);
  else if(t1=='Q')   output_high(pin_a4);
  else if(t1=='A')   output_low(pin_a4);
  else if(t1=='W')   output_high(pin_a5);
  else if(t1=='S')   output_low(pin_a5);

  else if(t1=='M')
  {

    t2=getc();
    if(t2 == '1')
      {
      t3=getc();
      ccsum_s1=getc();
        if(ccsum_s1 == ((t2 + t3)%255) )
        {

        n1=(float)t3/10;
        sprintf(isi1,"%3.1f",n1);
        glcd_rect(17,22,40,28,ON,OFF);
        glcd_text57(17, 22, isi1, 1, ON);
        bar1=n1/2;
        glcd_rect(7,20,13,42,ON,OFF); //kapatma
        glcd_bar(10,(42-bar1),10,42,7,ON);
        ccsum_ns1="    ";
        }
```

```
    else
    {
    ccsum_ns1="Nack";
    }
  }

if(t2 == '2')
  {
  t4=getc();
  ccsum_s2=getc();
    if(ccsum_s2 == ((t2 + t4)%255) )
    {

    n2=(float)t4/10;
    sprintf(isi2,"%3.1f",n2);
    glcd_rect(59,22,82,30,ON,OFF);
    glcd_text57(59, 22, isi2, 1, ON);
    bar2=n2/2;
    glcd_rect(49,22,55,42,ON,OFF); //kapatma
    glcd_bar(52,(42-bar2),52,42,7,ON);
    ccsum_ns2="    ";
    }
    else
    {
    ccsum_ns2="Nack";
    }
  }

if(t2 == '3')
  {
  t5=getc();
  ccsum_s3=getc();
    if(ccsum_s3 == ((t2 + t5)%255) )
    {

    n3=(float)t5/10;
    sprintf(isi3,"%3.1f",n3);
    glcd_rect(102,22,125,27,ON,OFF);
    glcd_text57(102, 22, isi3, 1, ON);
    bar3=n3/2;
    glcd_rect(94,20,97,42,ON,OFF); //kapatma
    glcd_bar(94,(42-bar3),94,42,7,ON);
    ccsum_ns3="    ";
    }
    else
    {
    ccsum_ns3="Nack";
```

```
        }
       }
  }


}

/********* ANA PROGRAM FONKSİYONU********/

void main ( )
{
   //65,5 ms overflow

  setup_ccp1(CCP_CAPTURE_RE);
  setup_ccp2(CCP_CAPTURE_RE);

    glcd_init(ON); // Grafik LCD hazırlanıyor ve ekran siliniyor

  write_eeprom(0,sd[0]);
  write_eeprom(1,sd[1]);
  write_eeprom(2,sd[2]);

  sd[0]=read_eeprom(0);
  sd[1]=read_eeprom(1);
  sd[2]=read_eeprom(2);

    glcd_text57(10, 3, yazi1, 1, ON);
    glcd_text57(55, 3, yazi2, 1, ON);
    glcd_text57(100, 3, yazi3, 1, ON);

    glcd_rect(0,0,127,63,NO,ON); // sıcaklık çerçece
        glcd_line(42, 1,42, 62, ON);
        glcd_line(84, 1,84, 62, ON);

    glcd_pixel(10,17,on);     // sıcaklık göstergesi
    glcd_line(8,18,12,18,ON);  // oda1
    glcd_rect(6,19,14,44,NO,ON);
    glcd_circle(10,52,8,ON,ON);

        glcd_pixel(52,17,on);     // sıcaklık göstergesi
    glcd_line(50,18,54,18,ON);      //oda2
    glcd_rect(48,19,56,44,NO,ON);
    glcd_circle(52,52,8,ON,ON);

        glcd_pixel(10,17,on);     // sıcaklık göstergesi
    glcd_line(92,18,96,18,ON);      //oda3
    glcd_rect(90,19,98,44,NO,ON);
```

```
    glcd_circle(94,52,8,ON,ON);

    glcd_text57(17, 13, sck1, 1, ON);
    glcd_line(17, 20,34, 20, ON);
    glcd_text57(17, 31, tlp1, 1, ON);
    glcd_line(17, 38,34, 38, ON);

    glcd_text57(59, 13, sck1, 1, ON);
    glcd_line(59, 20,76, 20, ON);
    glcd_text57(59, 31, tlp1, 1, ON);
    glcd_line(59, 38,76, 38, ON);

    glcd_text57(103, 13, sck1, 1, ON);
    glcd_line(103, 20,120, 20, ON);
    glcd_text57(103, 31, tlp1, 1, ON);
    glcd_line(103, 38,120, 38, ON);

  enable_interrupts(INT_CCP1);
  enable_interrupts(INT_CCP2);
  enable_interrupts(GLOBAL);
  clear_interrupt(INT_RDA);
  enable_interrupts(INT_RDA);
    CCP_1_HIGH=0x00;
    CCP_1_LOW=0x05;

    CCP_2_HIGH=0x00;
    CCP_2_LOW=0x0A;

//!   set_timer1(0);
    output_high(pin_a0);
    output_low(pin_a1);
    output_low(pin_a2);

  while(1) // Sonsuz döngü
  {
    if(input(DGS))
    {
      delay_ms(25);
      if(input(DGS))
      {
      i++;
      if(i==3) i=0;
      }

      if(i==0)
      {
      output_high(pin_a0);
```

```
      output_low(pin_a1);
      output_low(pin_a2);
      }

      if(i==1)
      {
      output_low(pin_a0);
      output_high(pin_a1);
      output_low(pin_a2);
      }

      if(i==2)
      {
      output_low(pin_a0);
      output_low(pin_a1);
      output_high(pin_a2);
      }
    }


    sprintf(ist1,"%3.1f",(float)sd[0]);
    sprintf(ist2,"%3.1f",(float)sd[1]);
    sprintf(ist3,"%3.1f",(float)sd[2]);
    glcd_rect(17,40,40,47,ON,OFF);
    glcd_text57(17, 40, ist1, 1, ON);
    glcd_rect(59,40,74,47,ON,OFF);
    glcd_text57(59, 40, ist2, 1, ON);
    glcd_rect(102,40,125,47,ON,OFF);
    glcd_text57(102, 40, ist3, 1, ON);

/*------------ODA 1----------------------*/
T_SLAVE1:
    output_high(gon);    // data göndermek için master mode açılıyor

    delay_ms(1);

    putc('1');
    putc('S');
    putc(sd[0]);
    ccsum1=(('1'+'S'+sd[0])%256);
    putc(ccsum1);
    delay_ms(10);
    clear_interrupt(INT_RDA);
    output_low(gon); // dinleyici moda geçiliyor

    delay_ms(30);
    if(t1 == 'X') {goto T_SLAVE1;}
```

```
R_SLAVE1:
    output_high(gon);

    delay_ms(1);



    putc('1');
    putc('D');
    putc('R');



    delay_ms(10);
    clear_interrupt(INT_RDA);
    output_low(gon);

    delay_ms(30);
    if (ccsum_ns1=="nack") {goto R_SLAVE1;}
 /*------------ODA 2----------------------*/
T_SLAVE2:
     output_high(gon);

    delay_ms(1);

    putc('2');
    putc('S');
    putc(sd[1]);
    ccsum2=(('2'+'S'+sd[1])%256);
    putc(ccsum2);
    delay_ms(10);
    clear_interrupt(INT_RDA);
    output_low(gon);

    delay_ms(30);
    if(t1=='Q') {goto T_SLAVE2;}
 R_SLAVE2:
    output_high(gon);

    delay_ms(1);



    putc('2');
    putc('D');
    putc('R');



    delay_ms(10);
    clear_interrupt(INT_RDA);
```

```
    output_low(gon);

    delay_ms(30);
    if (ccsum_ns2=="nack") {goto R_SLAVE2;}


   /*------------ODA 3----------------------*/
T_SLAVE3:
    output_high(gon);

    delay_ms(1);

    putc('3');
    putc('S');
    putc(sd[2]);
    ccsum3=(('3'+'S'+sd[2])%256);
    putc(ccsum3);     // ccsum ı önce yazarsan anlık hatayı veriyor
    delay_ms(10);
    clear_interrupt(INT_RDA);
    output_low(gon);

    delay_ms(30);
     if(t1=='W') {goto T_SLAVE3;}
R_SLAVE3:
    output_high(gon);

    delay_ms(1);


    putc('3');
    putc('D');
    putc('R');


    delay_ms(10);
    clear_interrupt(INT_RDA);
    output_low(gon);

    delay_ms(30);
    if (ccsum_ns3=="nack") {goto R_SLAVE3;}

  }
}
```

Slave CCS Kodları

Slave 1

```
#include <16f877a.h>

#device ADC=10  // 10 bitlik ADC kullanılacağı belirtiliyor.

// Denetleyici konfigürasyon ayarları
#fuses HS,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD

#use delay (clock=20000000) //osilatör frekansı belirtiliyor.

#use rs232 (baud=9600, xmit=pin_C6, rcv=pin_C7, parity=N, stop=1)
// RS232 protokolünün 9600 bit/sn baud hızında olacağını ve
// TX,RX uçlarının hangi pinler olacağını tanımlıyor
#define enable pin_C5
#include <kit_lcd.c>
  char t1,t2,t3,t4;
unsigned long int bilgi; // İşaretsiz 16 bitlik tam sayı tipinde değişken tanımlanıyor
float voltaj;
char ccsum1[];
char correct;

//********** ANA PROGRAM FONKSİYONU*******
#int_RDA
void seri()
{
  t1 = getc();
  if(t1=='1')
  {
    t2=getc();

    if(t2 == 'S')
    {
      t3 = getc();
      ccsum1=getc();
      if(ccsum1==((t1+t2+t3)%256))
      {
            delay_ms(10);
        output_high(enable);
        delay_ms(1);
      putc('T');
      delay_ms(10);
      output_low(enable);
```

```
        correct=t3;
        output_low(pin_d7);
        }
        else
        {
        delay_ms(10);
        output_high(enable);
        delay_ms(1);
        putc('X');
        output_high(pin_d7);
                delay_ms(10);
          output_low(enable);
        }

     }
     else if(t2=='D')
     {
          t4=getc();
         if(t4 == 'R')
         {
          delay_ms(10);
          output_high(enable);
          delay_ms(1);
          putc('M');
          putc('1');
          putc((char)voltaj);
          putc(('1'+ (char)voltaj)%255);
          delay_ms(10);
          output_low(enable);
         }

     }

  }

}


void main ( )
{
  setup_psp(PSP_DISABLED);       // PSP birimi devre dışı
  setup_timer_1(T1_DISABLED);    // T1 zamanlayıcısı devre dışı
  setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
  setup_CCP1(CCP_OFF);           // CCP1 birimi devre dışı
  setup_CCP2(CCP_OFF);           // CCP2 birimi devre dışı

  set_tris_a(0x01);  // RA0 Giriş olarak yönlendiriliyor
```

```
  setup_adc(adc_clock_div_32);   // ADC clock frekansı fosc/32
  setup_adc_ports(ALL_ANALOG); //RA0/AN0 girişi analog

  lcd_hazirla();   // LCD hazır hale getiriliyor

  set_adc_channel(0);   // RA0/AN0 ucundaki sinyal A/D işlemine tabi tutulacak
  delay_us(20);        // Kanal seçiminde sonra bu bekleme süresi verilmelidir

  printf(lcd_veri,"Sicaklik="); // LCD'ye yazı yazdırılıyor
  imlec(2,1);
  printf(lcd_veri,"istenen="); // LCD'ye yazı yazdırılıyor
  output_low(enable);
  enable_interrupts(GLOBAL);
  enable_interrupts(INT_RDA);
  while(true)   // sonsuz döngü
  {
    bilgi=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor

    voltaj=(0.0048828125*bilgi)*1000;   // Dijitale çevirme işlemine uğrayan sinyalin mV olarak gerilimi
hesaplanıyor
  // Her 10mV'ta 1 derece artma
    imlec(1,10); // İmleç 1. satır 10.sütunda
    printf(lcd_veri,"%3.1f'C ",(voltaj/10)); // LCD'ye sıcaklık değeri yazdırılıyor

  imlec(2,10);
  printf(lcd_veri,"%3.1u'C ",correct);

        if(correct<(voltaj/10 - 0.5))
        {
        output_high(pin_d0);
        output_low(pin_d1);
        output_low(pin_d2);
        output_low(pin_d3);
        }
        else if(correct>(voltaj/10-0.5) && correct<(voltaj/10+0.5) )
        {
        output_high(pin_d1);
        output_low(pin_d2);
        output_low(pin_d0);
        output_low(pin_d3);
        }
        else // (t3>(voltaj/10))
        {
        output_high(pin_d3); // ısıtıcı
```

```
        output_high(pin_d2); // yüksek sıcaklık
        output_low(pin_d1);  // istenen sıcaklık
        output_low(pin_d0);  // soğutucu
        }
  }
}
```

Slave 2

```
#include <16f877a.h>     // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.
#device ADC=10  // 10 bitlik ADC kullanılacağı belirtiliyor.
// Denetleyici konfigürasyon ayarları
#fuses HS,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD
#use delay (clock=20000000) // Gecikme fonksiyonu için kullanılacak osilatör frekansı belirtiliyor.

#use rs232 (baud=9600, xmit=pin_C6, rcv=pin_C7, parity=N, stop=1) // RS232 protokolünün 9600 bit/sn baud hızında olacağını ve
                                                  // TX,RX uçlarının hangi pinler olacağını tanımlıyor
#define enable pin_C5
#include <kit_lcd.c>
  char t1,t2,t3,t4;
unsigned long int bilgi; // İşaretsiz 16 bitlik tam sayı tipinde değişken tanımlanıyor
float voltaj;
char ccsum1[];
char correct;

//********** ANA PROGRAM FONKSİYONU*******
#int_RDA
void seri()
{
  t1 = getc();
  if(t1=='2')
  {
    t2=getc();

    if(t2 == 'S')
    {
      t3 = getc();
      ccsum1=getc();
      if(ccsum1==((t1+t2+t3)%256))
      {
            delay_ms(10);
        output_high(enable);
        delay_ms(1);
      putc('A');
      delay_ms(10);
      output_low(enable);
      correct=t3;
```

```
      output_low(pin_d7);
      }
      else
      {
      delay_ms(10);
        output_high(enable);
        delay_ms(1);
      putc('Q');
              delay_ms(10);
        output_low(enable);
      output_high(pin_d7);
      }


    }

    else if(t2=='D')
    {
        t4=getc();
       if(t4 == 'R')
       {
        delay_ms(10);
        output_high(enable);
        delay_ms(1);
        putc('M');
        putc('2');
        putc((char)voltaj);
        putc(('2'+ (char)voltaj)%255);
        delay_ms(10);
        output_low(enable);
       }
    }
  }
}
void main ( )
{
  setup_psp(PSP_DISABLED);       // PSP birimi devre dışı
  setup_timer_1(T1_DISABLED);    // T1 zamanlayıcısı devre dışı
  setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
  setup_CCP1(CCP_OFF);           // CCP1 birimi devre dışı
  setup_CCP2(CCP_OFF);           // CCP2 birimi devre dışı

  set_tris_a(0x01);  // RA0 Giriş olarak yönlendiriliyor



  setup_adc(adc_clock_div_32);   // ADC clock frekansı fosc/32
```

```
  setup_adc_ports(ALL_ANALOG); //RA0/AN0 girişi analog

  lcd_hazirla();   // LCD hazır hale getiriliyor
  set_adc_channel(0);   // RA0/AN0 ucundaki sinyal A/D işlemine tabi tutulacak
  delay_us(20);        // Kanal seçiminde sonra bu bekleme süresi verilmelidir
  printf(lcd_veri,"Sicaklik="); // LCD'ye yazı yazdırılıyor
  imlec(2,1);
  printf(lcd_veri,"istenen="); // LCD'ye yazı yazdırılıyor
  output_low(enable);
  enable_interrupts(GLOBAL);
  enable_interrupts(INT_RDA);
  while(true)   // sonsuz döngü
  {
    bilgi=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor

    voltaj=(0.0048828125*bilgi)*1000;   // Dijitale çevirme işlemine uğrayan sinyalin mV olarak gerilimi
hesaplanıyor
  // Her 10mV'ta 1 derece artma
      imlec(1,10); // İmleç 1. satır 10.sütunda
    printf(lcd_veri,"%3.1f'C ",(voltaj/10)); // LCD'ye sıcaklık değeri yazdırılıyor

  imlec(2,10);
  printf(lcd_veri,"%3.1u'C ",correct);

        if(correct<(voltaj/10 - 0.5))
        {
        output_high(pin_d0);
        output_low(pin_d1);
        output_low(pin_d2);
        output_low(pin_d3);
        }
        else if(correct>(voltaj/10-0.5) && correct<(voltaj/10+0.5) )
        {
        output_high(pin_d1);
        output_low(pin_d2);
        output_low(pin_d0);
        output_low(pin_d3);
        }
        else // (t3>(voltaj/10))
        {
        output_high(pin_d3); // ısıtıcı
        output_high(pin_d2); // yüksek sıcaklık
        output_low(pin_d1);  // istenen sıcaklık
        output_low(pin_d0);  // soğutucu
        }
  }
}
```

Slave 3

```
#include <16f877a.h>     // Kullanılacak denetleyicinin başlık dosyası tanıtılıyor.

#device ADC=10  // 10 bitlik ADC kullanılacağı belirtiliyor.

// Denetleyici konfigürasyon ayarları
#fuses HS,NOWDT,NOPROTECT,NOBROWNOUT,NOLVP,NOPUT,NOWRT,NODEBUG,NOCPD

#use delay (clock=20000000) // Gecikme fonksiyonu için kullanılacak osilatör frekansı belirtiliyor.


#use rs232 (baud=9600, xmit=pin_C6, rcv=pin_C7, parity=N, stop=1) // RS232 protokolünün 9600 bit/sn baud hızında olacağını ve
                                                  // TX,RX uçlarının hangi pinler olacağını tanımlıyor
#define enable pin_C5
#include <kit_lcd.c>
  char t1,t2,t3,t4;
unsigned long int bilgi; // İşaretsiz 16 bitlik tam sayı tipinde değişken tanımlanıyor
float voltaj;
char ccsum1[];
char correct;

//********** ANA PROGRAM FONKSİYONU*******
#int_RDA
void seri()
{
  t1 = getc();
  if(t1=='3')
  {
    t2=getc();

    if(t2 == 'S')
    {
      t3 = getc();
      ccsum1=getc();
      if(ccsum1==((t1+t2+t3)%256))
      {
            delay_ms(10);
        output_high(enable);
        delay_ms(1);
      putc('S');
      delay_ms(10);
      output_low(enable);
      correct=t3;
      output_low(pin_d7);
      }
      else
```

```
      {
      delay_ms(10);
        output_high(enable);
        delay_ms(1);
      putc('W');
      output_high(pin_d7);
              delay_ms(10);
        output_low(enable);
      }
    }
      else if(t2=='D')
    {
        t4=getc();
       if(t4 == 'R')
       {
        delay_ms(10);
        output_high(enable);
        delay_ms(1);
        putc('M');
        putc('3');
        putc((char)voltaj);
        putc(('3'+ (char)voltaj)%255);
        delay_ms(10);
        output_low(enable);
       }

    }
  }
}


void main ( )
{
  setup_psp(PSP_DISABLED);       // PSP birimi devre dışı
  setup_timer_1(T1_DISABLED);    // T1 zamanlayıcısı devre dışı
  setup_timer_2(T2_DISABLED,0,1); // T2 zamanlayıcısı devre dışı
  setup_CCP1(CCP_OFF);           // CCP1 birimi devre dışı
  setup_CCP2(CCP_OFF);           // CCP2 birimi devre dışı

  set_tris_a(0x01);  // RA0 Giriş olarak yönlendiriliyor



  setup_adc(adc_clock_div_32);   // ADC clock frekansı fosc/32
  setup_adc_ports(ALL_ANALOG); //RA0/AN0 girişi analog

  lcd_hazirla();   // LCD hazır hale getiriliyor
```

```
  set_adc_channel(0);   // RA0/AN0 ucundaki sinyal A/D işlemine tabi tutulacak
  delay_us(20);         // Kanal seçiminde sonra bu bekleme süresi verilmelidir

  printf(lcd_veri,"Sicaklik="); // LCD'ye yazı yazdırılıyor
  imlec(2,1);
  printf(lcd_veri,"istenen="); // LCD'ye yazı yazdırılıyor
  output_low(enable);
  enable_interrupts(GLOBAL);
  enable_interrupts(INT_RDA);
  while(true)   // sonsuz döngü
  {
    bilgi=read_adc();  // ADC sonucu okunuyor ve bilgi değişkenine aktarılıyor

    voltaj=(0.0048828125*bilgi)*1000;   // Dijitale çevirme işlemine uğrayan sinyalin mV olarak gerilimi
hesaplanıyor
  // Her 10mV'ta 1 derece artma
      imlec(1,10); // İmleç 1. satır 10.sütunda
    printf(lcd_veri,"%3.1f'C ",(voltaj/10)); // LCD'ye sıcaklık değeri yazdırılıyor

  imlec(2,10);
  printf(lcd_veri,"%3.1u'C ",correct);

        if(correct<(voltaj/10 - 0.5))
        {
        output_high(pin_d0);
        output_low(pin_d1);
        output_low(pin_d2);
        output_low(pin_d3);
        }
        else if(correct>(voltaj/10-0.5) && correct<(voltaj/10+0.5) )
        {
        output_high(pin_d1);
        output_low(pin_d2);
        output_low(pin_d0);
        output_low(pin_d3);
        }
        else // (t3>(voltaj/10))
        {
        output_high(pin_d3); // ısıtıcı
        output_high(pin_d2); // yüksek sıcaklık
        output_low(pin_d1);  // istenen sıcaklık
        output_low(pin_d0);  // soğutucu
        }
  }
}
```