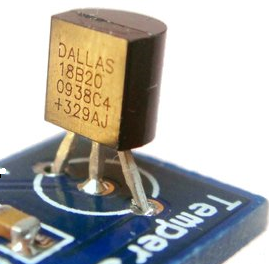
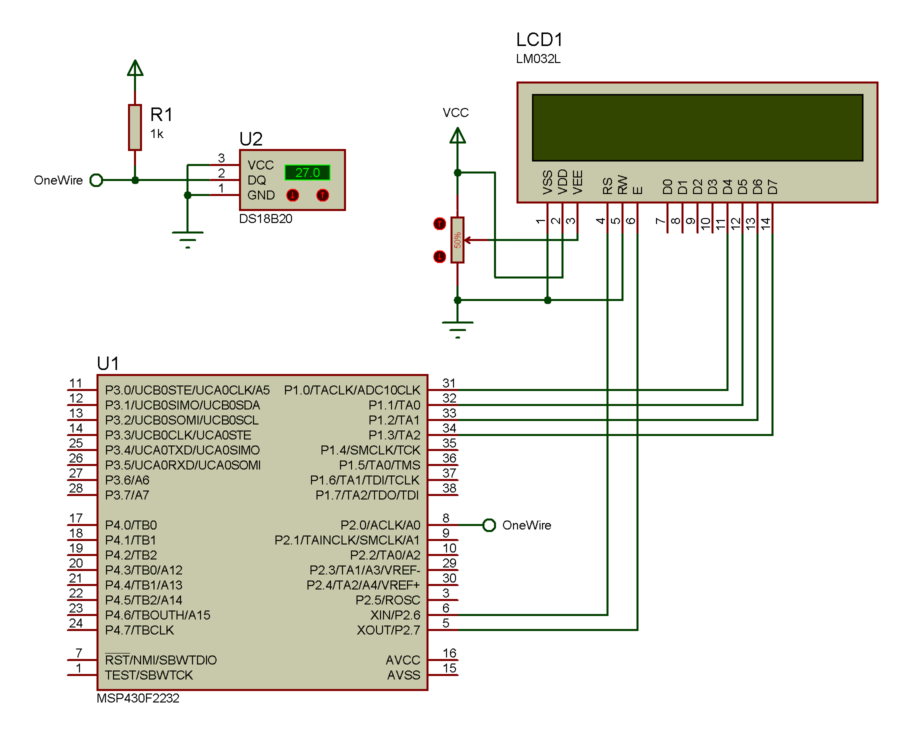
**MSP430 Uyg.34 – DS18B20 Sıcaklık Sensörü Uygulaması**

Ferudun GÖKCEGÖZ, 02 Eylül 2011, Cuma

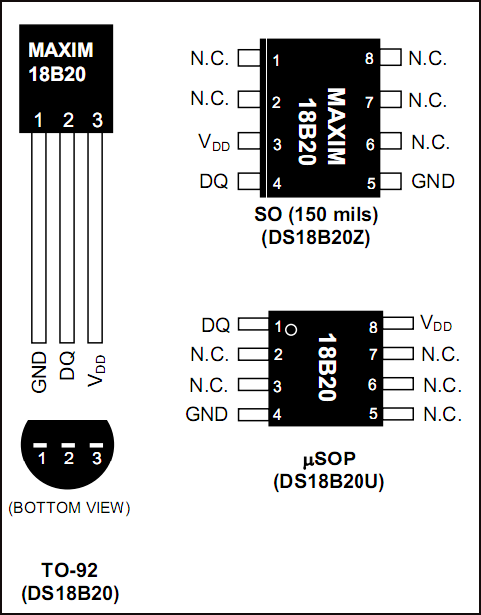


Merhaba arkadaşlar. Yeni bir uygulamamızda yine sizlerle birlikteyiz. Geçenki uygulamada söylediğimiz gibi bu dersimizde OneWire protokolü ile haberleşen Maxim – Dallas firmasına ait DS18B20 sıcaklık sensörü ve termostat entegresi ile bir uygulama yapacağız. Bir önceki yazımda OneWire protokolünden bahsetmiştim. Bu nedenle bu yazıda OneWire protokolünden bahsetmeyip, sadece aynı C kütüphanesini kullanarak, DS18B20 ile nasıl haberleşeceğimizi anlatacağım. İlk olarak devre şemasını vererek uygulamamıza başlayabiliriz.

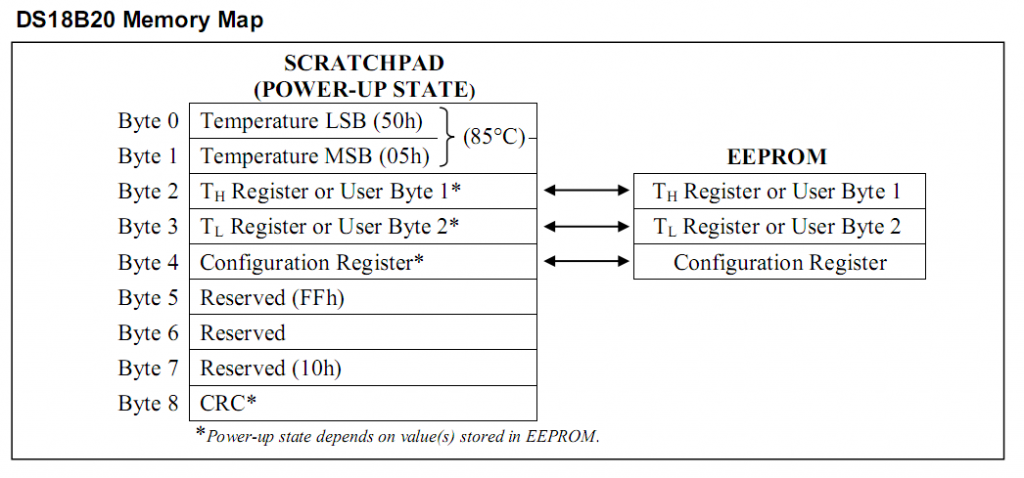


Devre şemasında görüldüğü gibi bir adet MSP430, bir adet LCD ekran ve DS18B20 entegremiz bulunmaktadır. İlk olarak DS18B20 entegresinden bahsedelim isterseniz.

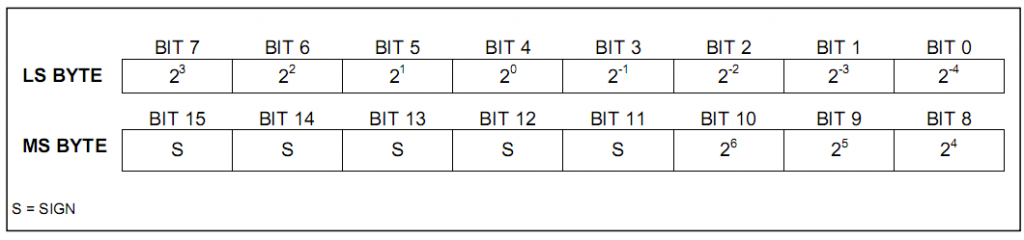
* DS18B20 OneWire protokolüyle haberleşebilen dolayısıyla mikrodenetleyici ile haberleşmek için sadece tek bir hattın yeterli olduğu bir entegredir. Ekstradan besleme gerektirmez fakat harici olarak besleme girişide bulunmaktadır.
* OneWire cihazlarına özgü olan 64 bit lik Rom Code bulundurmaktadır. Önceki yazımızda bahsettiğimiz gibi, bu 64 bitin ilk 8 biti Aile Kodu, Sonraki 48 biti seri numarası, son 8 biti ise CRC kodudur.
* Harici besleme kullanılacaksa 3V ~5.5V arasında bir besleme aralığına sahiptir. Maksimum 750 msn de 12 bitlik çözünürlüğe sahip (8bit tam kısım + 4 bit ondalık kısım olmak üzere) sıcaklık verisi elde edilebilir.
* Standart olarak 12 bitlik çözünürlüğe sahip olmasına rağmen 9,10 ve 11 bit çözünürlükte çalışmak üzerede ayarlanabilir. Çözünürlük sayısı azaldıkça, sıcaklık verisinin 8 biti sabit kalmakta sadece 4 bitlik ondalık kısmın bit sayısı azalmaktadır.
* -55 C ~ +125 C(-67 F ~ +257 F) arasında ölçüm aralığına sahip olup -10 C ~+85 C arasında 0.5 C hassasiyetle ölçüm yapılabilmektedir.
* Termostat olarak kullanmak için kullanıcının tanımlayabileceği alarm kaydedicileri barındırmaktadır. Entegrenin ölçtüğü sıcaklık ile alarm kaydedicilerine kaydedilen sıcaklık otomatik olarak karşılaştırılıp alarm durumu oluşup oluşmadığı entegreye alarm sorgulama komutu gönderilerek tespit edilebilir.
* TO92, SO8, uSOP kılıflarında üretilebilmektedir. Aşağıda entegrenin kılıflarına ait görüntüleri görebiliriz.



Aşağıda DS18B20 içerisinde bulunan kaydedicilere ait RAM Haritası görülmektedir. Şimdi bunlardan önemli olanlarına kısaca değinelim.

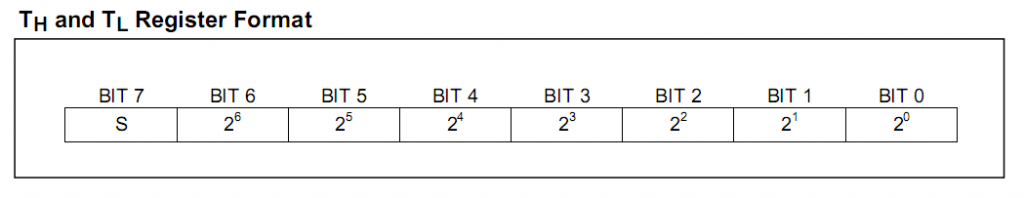


**TEMPERATURE KAYDEDİCİLERİ**



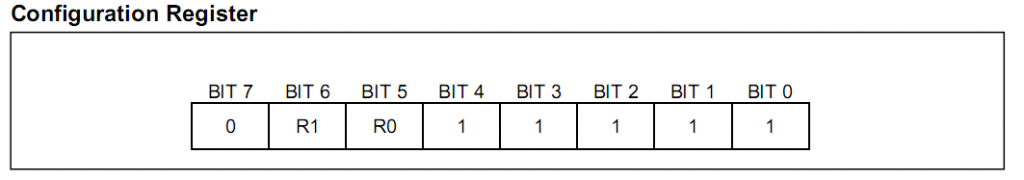
Yukarıdaki tablodanda görüldüğü gibi sıcaklık verisini tutan kaydedicilerdir. Low Byte ve High Byte olmak üzere iki byte tır. Low Byte ın ilk 4 biti ondalık kısma ait datayı tutan bitlerdir. Örneğin ilk 4 bit “1000″ ise bu şu anlama gelmektedir. BIT3 ün byte üzerindeki ağırlığı 0.5 olduğu için, sıcaklık veriside “XX.5″ şeklinde olacaktır. Sırasıyla BIT0,BIT1,BIT2 ve BIT3 ün ağırlıkları 0.0625, 0.125, 0.25, 0.5 olmaktadır. Yazılım içerisindeki sıcaklığın ondalık kısmınıda görmek istiyorsak bu bitleri kontrol edip düzgün bir şekilde lcd ekrana basmalıyız. Low Byte ın son 4 biti ise sıcaklık verisinin tam kısmının low nibble ı yani düşük dört bitidir. Sıcaklık verisinin yüksek dört biti ise High Byte ın ilk dört bitidir. Yazılım içerisinde sıcaklık verisi elde edilirken, Low Byte ın son dört biti ile High Byte ın ilk dört biti bir araya getirilerek 8 bitlik bir değer(sıcaklık verisinin tam kısmı) elde edilir. High Byte ın son dört biti ise işaret bitidir. Ölçülen sıcaklık negatif ise bu dört bit 1 olmaktadır. Ve sıcaklık verisi ise 2 ye tümleyen formatta bu kaydedicilerden okunmaktadır. Bu nedenle negatif sıcaklık ölçümü yapılacaksa bu duruma dikkat edilmelidir.

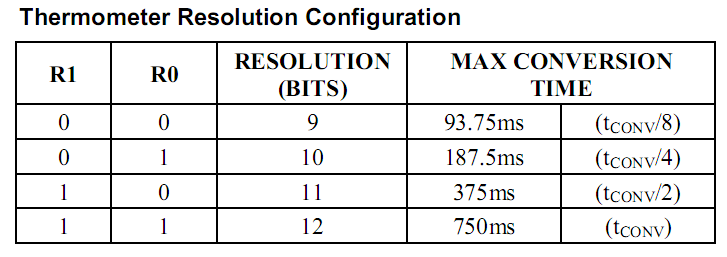
**TH ve TL KAYDEDİCİLERİ**



Bu kaydediciler 8 bitlik iki adet register dır. Ve entegre termostat olarak kullanılmak isteniyorsa bu kaydedicilere yüklenen değerlerle ölçülen sıcaklık değeri kıyaslanıp alarm durumu olup olmadığı tespit edilebilir.  Bu kaydedicilerin son biti ise yine işaret bitidir. Ve bu kaydedicilerin değerleri aynı isimde EEPROM hafızada tutulmaktadır. Power On durumunda ise ScratchPAD deki karşılıklarının değeri EEPROM daki değerlerdir. Eğer entegrenin termostat özelliği kullanılmayacak ise bu kaydediciler kullanıcı tarafından genel kullanım amacı ile kullanılabilir.

**CONFIGURATION KAYDEDİCİSİ**





ScratchPad in 4. byte ı ise Configuration Register ı dır. Bu kaydedici ile de entegrenin çözünürlüğü ayarlanmaktadır. Yazının başında belirttiğimiz gibi entegremizin çözünürlüğü default olarak 12 bit olmasına rağmen 9,10 ve 11 bit olarakta kullanılabilmektedir. Bu ayarda yukarıda görüldüğü gibi konfigürasyon kaydedicisinin R0 ve R1 bitleri ile ayarlanmaktadır. Tablodan görüldüğü gibi çözünürlük düştükçe sıcaklık verisini elde etmek için geçen çevrim süreside düşmektedir. Ayrıca bu kısımda belirtilmesi gereken birşey varki, çözünürlük düşürüldüğünde eksilen bitler sıcaklık verisinin tam kısmından değil ondalık kısmındaki bitlerden düşürülmektedir. Buda kullanıcı için gayet güzel bir özelliktir.

ScratchPad in 5, 6 ve 7. byte ı reserve edilmiştir. Bu bitler kullanılmamaktadır. 8. byte ı ise CRC kodudur. Bir önceki yazımızda belirttiğimiz gibi CRC kodu bir hata kontrol algoritmasından geçirildikten sonra elde edilen bir veridir. Ve okunan verinin doğruluğunu test etmek için oldukça iyi bir yöntemdir. Birkaç byte veri iletiminde çok önemli olmasada onlarca bytelık veri paketlerinin iletildiği bir haberleşme protokolünde hata kontrol algoritmaları (CRC, LRC vb.) oldukça önem taşımaktadır.

Bir önceki yazımızda belirtiğimiz gibi OneWire cihazlarla haberleşebilmek için fonksiyon kodları bulunmaktadır. Ve yapılan işlem bu fonksiyon kodları ile belirlenmektedir. DS18B20 entegresi için rom code için ve scratchpad için birtakım fonksiyon kodları bulunmaktadır. Bunlardan da bahsedecek olursak,

**ROM Komutları**

1. **Search ROM (0xF0)**
2. **Read ROM (0×33)**
3. **Match ROM (0×55)**
4. **Skip ROM (0xCC)**
5. **Alarm Search (0xEC)**

**DS18B20 Fonksiyon Komutları**

1. **Convert T (0×44)**
2. **Write ScratchPad (0x4E)**
3. **Read ScratchPad (0xBE)**
4. **Copy ScratchPad (0×48)**
5. **Recall E2 (0xB8)**
6. **Read Power Supply (0xB4)**

DS18B20 ile haberleşebilmek için datalara ait bir veri sekansı bulunmaktadır. İlk olarak hat resetlenip presence pulse beklenir. Ve hatta bir cihaz bulunduğu ve hattaki cihazın haberleşmeye uygun olduğu anlaşılır. Daha sonra ROM komutu gönderilir. Bu aşamada match rom(0×55) komutu kullanılarak cihazın seri numarasının okunması suretiyle hatta birden fazla cihaz varsa tanımlama işi gerçekleştirilebilir. Rom komutu gönderilmeyecek ise skip rom(0xCC) komutu gönderilerek bu ikinci aşama geçilir. Üçüncü aşama olarakta fonksiyon kodlarından biri gönderilerek yapılacak işlem ne ise o yapılır. Örneğin sıcaklık verisinin okunması için bir çevrim başlatılmasını istiyorsak convert t (0×44) komutu göndererek sıcaklık verisini okuyabiliriz.

Daha detaylı bilgi almak isteyenler DS18B20 ye ait datasheet i [**buradan**](Dokumanlar/DS18B20.pdf) inceleyebilirler. Birde şunu söylemeliyimki yine bu ürüne çok benzer bir şekilde, DS18S20 entegresi bulunmaktadır. O entegrede aynen DS18B20 gibidir. Farklılık olarak örneğin çözünürlüğü ayarlanamakta ve standart olarak 9 bittir.. O entegre kullanılarakta 9 bit sıcaklık verisi elde edilebilir. Detaylı bilgi için DS18S20 ye ait datasheet [**buradan**](Dokumanlar/DS18S20.pdf) incelenebilir.

Gelelim uygulamaya ait yazılımımıza…

|  |  |
| --- | --- |
| **[main.c](http://www.mcu-turkey.com/msp430-uyg-34/" \l "codesyntax_1" \o "Click to show/hide code block)** | **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/code.png](http://www.mcu-turkey.com/msp430-uyg-34/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/printer.png](http://www.mcu-turkey.com/msp430-uyg-34/#codesyntax_1)** **[http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/themes/default/images/info.gif](http://www.mcu-turkey.com/wp-content/plugins/wp-synhighlight/About.html)** |

#include "io430.h"

#include "in430.h"

#include "stdbool.h"

#include "lcd\_4bit.h"

#include "OneWire.h"

bool negative=**false**;

bool limited=**false**;

void LCD\_Write\_Int (unsigned int val,char mul)

{

char j;

int temp=1;

unsigned char str[4];

for(j=0;j<mul;j++)

{str[j] = ((val/temp)%10+48); temp\*=10;}

for(j=0;j<mul;j++)

lcd\_putch(str[mul-1-j]);

}

void Read\_DS18B20 (unsigned char \*tam\_adr, unsigned int \*ondalik\_adr)

{

unsigned char msb,lsb,temp;

\*tam\_adr = 0;

\*ondalik\_adr = 0;

while(Ow\_Reset());

Ow\_WriteByte(0xCC);

Ow\_WriteByte(0x44);

\_\_delay\_cycles(120);

while(Ow\_Reset());

Ow\_WriteByte(0xCC);

Ow\_WriteByte(0xBE);

\_\_delay\_cycles(800000);

lsb = Ow\_ReadByte();

msb = Ow\_ReadByte();

if(msb & 0xF0) negative = **true**;

\*tam\_adr=(msb<<4)+(lsb>>4);

if(\*tam\_adr>99) limited = **true**;

temp = lsb & 0x0F;

if(temp & 0x01) \*ondalik\_adr += 625;

if(temp & 0x02) \*ondalik\_adr += 1250;

if(temp & 0x04) \*ondalik\_adr += 2500;

if(temp & 0x08) \*ondalik\_adr += 5000;

}

void main (void)

{

unsigned char TAM\_KISIM = 0;

unsigned int ONDALIK\_KISIM = 0;

WDTCTL = WDTPW + WDTHOLD;

DCOCTL=CALDCO\_1MHZ;

BCSCTL1=CALBC1\_1MHZ;

P1SEL = 0x00;

P1OUT = 0x00;

P1DIR = 0xFF;

P2SEL = 0x00;

P2OUT = 0x00;

P2DIR = 0xFF;

lcd\_init();

lcd\_goto(1,1);

lcd\_puts(" DS18B20 TERMOMETRE ");

lcd\_goto(2,1);

lcd\_puts(" UYGULAMASI ");

\_\_delay\_cycles(1000000);

lcd\_puts("");

for(;;)

{

Read\_DS18B20(&TAM\_KISIM,&ONDALIK\_KISIM);

if(limited==**false** && negative==**false**)

{

lcd\_clear();

lcd\_goto(1,1);

lcd\_puts(" SICAKLIK ");

lcd\_goto(2,6);

LCD\_Write\_Int(TAM\_KISIM,2);

lcd\_putch('.');

LCD\_Write\_Int(ONDALIK\_KISIM,4);

lcd\_putch(' ');

lcd\_putch(223);

lcd\_putch('C');

}

else

{

lcd\_clear();

lcd\_goto(1,1);

lcd\_puts(" OLCULEN SICAKLIK ");

lcd\_goto(2,1);

lcd\_puts(" ARALIK DISI ");

negative = **false**;

limited = **false**;

}

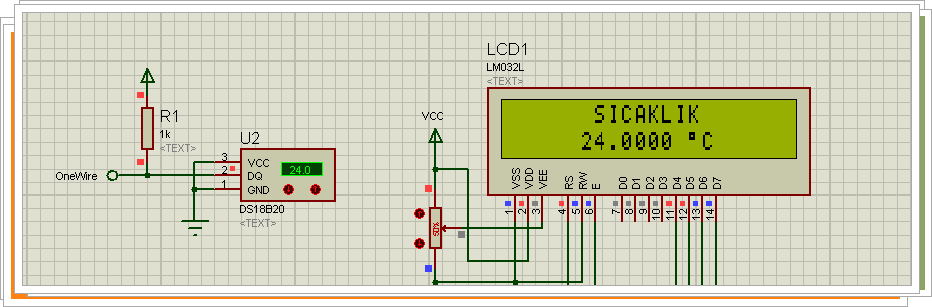
\_\_delay\_cycles(100000);

}

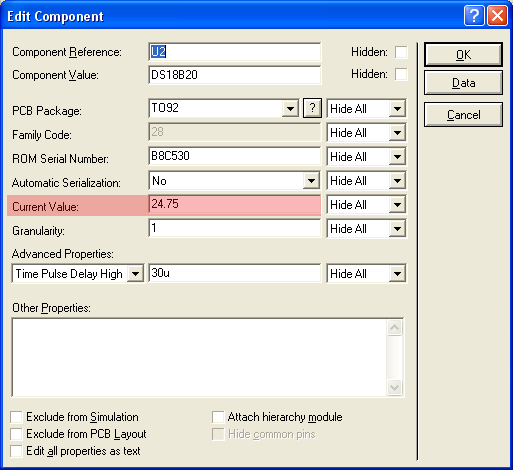
}

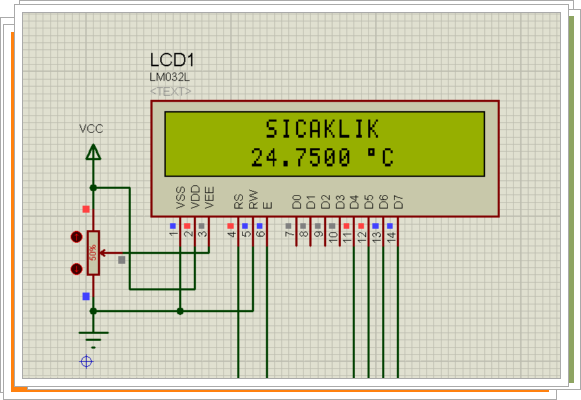
Yazılımda kullanılan OneWire.c ve OneWire.h kütüphane dosyaları aynı. O yüzden tekrar vermek istemedim. Uygulamayı proteus üzerinde kurmak isteyen arkadaşlar bir önceki yazıya bakabilirler…

Yazılımdan kısaca bahsedecek olursak, ilk olarak lcd ekranda “DS18B20 TERMOMETRE UYGULAMASI” string i yazdırılmaktadır. Daha sonra sonsuz döngü içerisinde sensör den sıcaklık verisi tam kısım ve ondalık kısım olarak okunup lcd ekrana yazdırılmaktadır. Okunan sıcaklık 0~99 C arasında değilse lcd ekranda “OLCULEN SICAKLIK ARALIK DISI” yazısı yazdırılmaktadır. Ardından da 100 ms gecikme verilip aynı işlemler sürekli olarak tekrarlanmaktadır. Uygulamaya ait ekran görüntüsünüde verelim.



*isterseniz entegrenin ayarlarından ondalık sıcaklık belirleyip o şekilde de ölçüm yapabilirsiniz.*





Görüldüğü gibi başarılı bir şekilde sıcaklık ölçümü gerçekleştirilmiştir. Ben bu uygulamayı PIC üzerinde de gerçekleştirdim. Bu tür uygulamalarda en çok dikkat edilmesi gereken mesele gecikme üreten fonksiyonların doğruluğu. O yüzden bu uygulamayı launchpad üzerinde denerken, dahili RC osilatörün stabil clock darbeleri üretememesi nedeniyle problem çıkabilir. Bu nedenle sinyaller dikkatlice osiloskopta incelenip, msp430 üzerinde o şekilde uygulama yapılması tavsiye edilir.

Geldik bir yazımızın daha sonuna. Bir sonraki yazımda LaunchPad üzerinde RC Servo Motor uygulaması yapacağız. Uygulama hazır. Muhtemelen birkaç gün içerisinde siteye eklerim. Bir sonraki yazımda görüşmek dileğiyle.

Hadi bakalım kalın sağlıcakla…

**Ferudun GÖKCEGÖZ**

**fgokcegoz@yahoo.com**