Modül 10: MSP430 ile SD Kart Kullanımı ve Uygulamaları

Giriş

Bilginin depolanması elektronik devre ve sistemlerde de önemli bir konudur. Elde edilen verilerin enerji kesintilerine bağımlı olmaksızın kalıcı olarak saklanmak istenebilir. Bu gibi durumlarda verileri kalıcı olarak saklayan hafıza elemanları kullanılır.

Elektronik sistemlerde veriler çoğunlukla sayısal olarak saklanır. Saklanan verinin boyutuna ve uygulama özelliklerine göre çeşitli saklama elemanları kullanılabilir. Bunlardan bazıları rom, eprom, eeprom, flash, flash disk, HDD, SD kart, MMC kart, eMMC kart gibi saklama elemanlarıdır. Rom, eprom gibi yapılar teknolojileri eski olduğu için kullanım alanları azalmıştır. Benzer şekilde flash, eeprom gibi hafıza elemanları küçük, orta ölçekli mikrodenetleyicili sistemlerde yaygın olarak kullanılmaktadır. Büyük ölçekli elektronik sistemlerde ise yüksek boyutta verilerin depolanması gerekebilir. Bu gibi durumlarda flash disk, HDD, SD kart, MMC kart, eMMC kart gibi yüksek kapasiteli saklama elemanları kullanılır. Örneğin fotoğraf makinelerinde SD kartlar yaygın olarak kullanılmaktadır. Benzer şekilde kişisel bilgisayarlarda ise verileri saklamak için çoğunlukla HDD kullanılır.

Bu şekilde kullandığınız sistemin ihtiyacına göre uygun saklama elamanını kullanabilirsiniz. Bu modül kapsamında SD kartlar hakkında bilgi verilip yapıları incelenecektir. MSP430 geliştirme kartı üzerinde SD kart kullanım olanağı olduğundan MSP430 denetleyiciler ile SD kart kullanımı uygulamaları gerçekleştirilecektir.

SD Kartlar

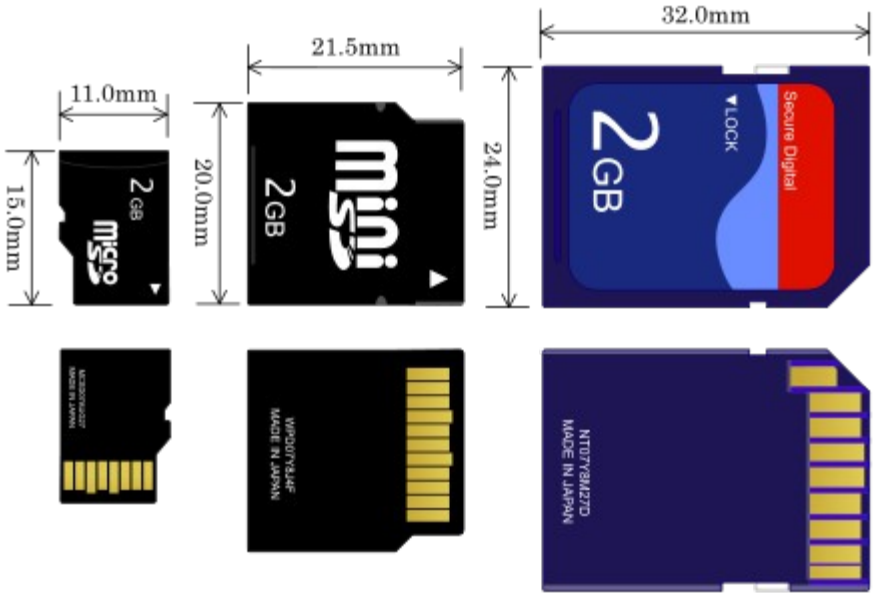
SD (Secure Digital/Güvenli Sayısal) kartlar, SD Kart derneği(www.sdcard.org) tarafından geliştirilen kalıcı hafızaya sahip hafıza kartı yapısıdır. SD kartlar flash hafıza temellidir ve dijital kamelarda, mobil telefonlarda, e-kitap okuyucularda, tablet bilgisayarlarda, ortam oynatıcılarda, oyun konsollarında vb. bir çok üründe yaygın olarak kullanılmaktaıdır.

Şekil 1'de SD kartların başlangıcından günümüze kadar ki kullanılan kart tipleri ve özellikleri görülmektedir.

Kart Türü	Boyutları
SD, SDHC, SDXC, SDIO	32 x 24 x 2.1 mm
miniSD, miniSDHC, mini SDIO	21.5 x 20 x 1.4 mm
microSD, microSDHC, micro SDXC	15 x 11 x 1.0 mm

Kart Türü	Kabul Yılı	Azami Boyut	Yazma Hızı	FAT Türü
SD	2000	4GB	0.9 - 20MB/s	FAT16
SDHC	2006	32GB	2 - 40MB/s	FAT32
SDXC	2009	2TB	max 300MB/s	exFAT

Şekil 1: SD Kart Özellikleri



Şekil 2: Temsili SD kart Görüntüleri

Şekil 2'de SD kartların temsili görüntüleri görülmektedir. Piyasada bir çok firma SD kart üretmektedir. Ölçüler standart olduğu için üretici firmalar bu ölçülere sadık kalmak durumundadırlar.

SD Kartların Kullanımı

SD kartlar ile haberleşebilmek, veri okuyup yazabilmek için çeşitli haberleşme protokolleri vardır. SD kartlar en az 3 haberleşme protokolünü desteklemektedir. Yaygın kullanılan iletişim protokolleri aşağıda ki gibidir.

1-Bit SD: Komut ve veri yolunun ayrı olduğu özel iletişim protokolüdür.

4-Bit SD: Haberleşme için 4-Bit veri yolu, ayrıca ek olarak kontrol ve komut işaretleri kullanılır. SD kartlarda en yaygın olarak kullanılan haberleşme protokolüdür.

SPI: Serial Peripheral Interface yani Seri Çevresel Arayüz standart hale gelmiş seri senkron haberleşme protokolüdür. SD kartlar standart SPI protokolü ile haberleşebilirler. Gömülü sistemlerde ve kaynakları kısıtlı, küçük ölçekli sistemlerde SD kart ile haberleşebilmek için çoğunlukla SPI protokolü kullanılır.

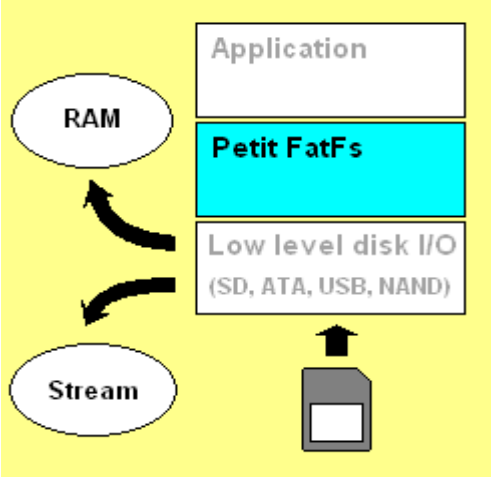
SD kartlar ile haberleşmek için iletişim protokolünü belirledikten sonra gerekli işlemler yapılarak okuma/yazma işlemi yapılabilir. Temel olarak bilinmesi gereken şey SD kartlara 512 byte bloklar halinde veri yazma yada okuma yapılabilir. SD kartların iç yapısı kontrol işlemleri başlı başına bir konu olduğu için modülde bu kadarıyla yetinilecektir. Detaylı bilgi www.sdcard.org adresinden edinilebilir.

SD kartlar bilgisayarlar, mobil cihazlar, fotoğraf makineleri, video cihazları vs. bir çok cihazlarda kullanıldığı için SD kartların içerisine gelişi güzel veri yazarsak yazdığımız verileri diğer sistemlere aktaramayız. Örneğin mikrodenetleyici yada benzeri bir denetleyici ile SD karta bir kaç kb boyutunda bir veri yazdığımız varsayalım. Sonrasında SD kartı bilgisayara taktığımızda işetim sistemi muhtemelen SD kartın içerisine ulaşamayarak kartın biçimlendirilmesini söylecektir. Bunun nedeni SD kartın içerisinde FAT(File Allocation Table) dosyalama sisteminin olmayışıdır. Bu nedenle SD karta bir ver yani dosya yazmak için FAT dosyalama sistemi kullanılmalıdır.

Dosyalama sistemleri, işletim sistemleri ve diğer depolama elemanları ile ortak bir şekilde veri okuma yazmak için kullanılan sistemlerdir. Örneğin fotoğraf makinesi ile çekilip SD karta kayıt edilen bir resmin bilgisayarda görüntülenebilmesi için fotoğraf makinesinin çekilen resmi işetim sistemin desteklediği NTFS, FAT16, FAT32 gibi dosyalama sistemlerine uygun olarak SD karta kayıt etmesi gerekir. Benzer şekilde bizde MSP430 ile yaptığımız uygulamalarda SD karta yazdığımız, okudğumuz verilerin işletim sistemi ile uyumlu olmasını istiyorsak dosyalama sistemi kullanmalıyız.

Petit FAT Kütüphanesi

Mikrodenetleyiciler ile dosya sistemi kullanarak SD karta veri yazmak, okumak için öncelikle SD kartın gerekli kütüphanelerinin yazılması sonrasında FAT dosya sistemi ile ilgili kütüphanelerin yazılması gerekmektedir. Bu işlemler tecrübe isteyen karmaşık konular olduğu için genellikle hazır kütüphaneler kullanılarak SD kart işlemleri yapılır. İnternet üzerinde yaygın olarak bilinen ve kullanılan FAT kütüphaneleri Elm Chan (http://elm-chan.org) tarafından hazırlanan FatFs ve Petit FAT kütüphaneleridir. Kullanımı ücretsiz ve açık kodlu olan bu (C ile yazılmış) kütüphaneleri kolayca kullandığınız mikrodenetleyici için uyarlayıp kullanabilirsiniz. Petit FAT kütüphanesi FatFs kütüphanesinin özellikleri kısıtlanmış versiyonudur. Özellikleri kısıtlanmıştır fakat sistem gereksinimleri daha azdır. Ayrıca bu kütüphaneler SD kart haricinde çeşitli taşınabilir hafıza cihazları ile birlikte kullanılabilir. Detaylı bilgi http://elm-chan.org/ adresinden edinilebilir.



Şekil 3: Petit FatFs Kütüphanesi Uygulama Katmanları

Şekil 3'te Petif FatFs kütüphanesinin uygulama katmanları görülmektedir. Görüldüğü gibi en alt katmanda SD kart yada diğer hafıza elemanları için gerekli alt seviye kütüphaneler bulunmaktadır. Bu katmanın üzerinde ise Petit FatFs kütüphanemiz yer almaktadır. En üst katmanda ise kendi uygulama kodlarımız bulunmaktadır.

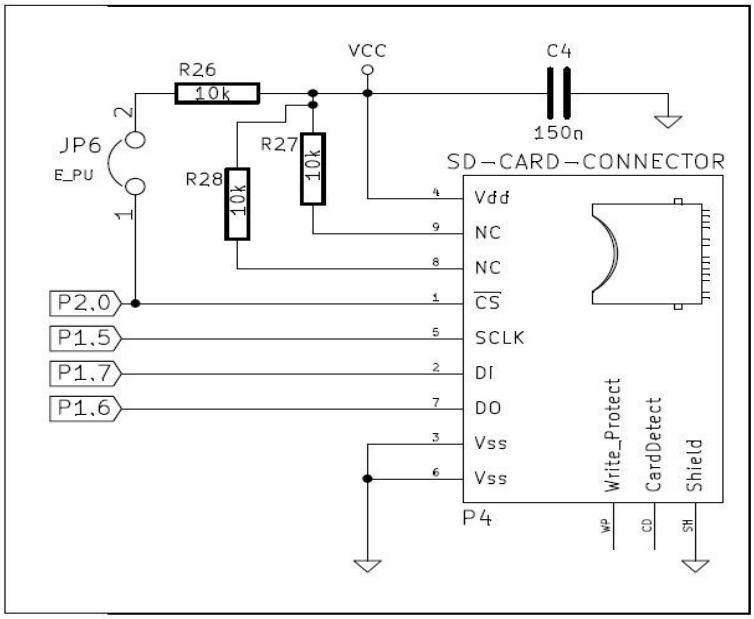
Petit FatFs kütüphanesi toplamda yaklaşık olarak 2K-4K byte kod hafızası 44 byte RAM hafızası kullanmaktadır. Ek olarak stack(yığın) hafızayı kullanmaktadır. FatFs kütüphanesi ise yaklaşık 16K byte kod hafıza en az 1K byte RAM gerektirmektedir. Bu nedenle MSP430G2553 denetleyicisi ile SD kart uygulamalarında Petit FatFs kütüphanesini kullanacağız.

Petit FatFs kütüphanesinin sistem gereksinimleri düşük olmasından dolayı bazı kısıtlamaları vardır. Petit FatFs kütüphanesi ile Dosya oluşturma yapılamaz ve var olan dosyanın boyutu değiştirilemez. Ayrıca dosyanın zaman bilgileri de değiştirilemez. Sadece kart içinde bulunan dosyalar okunabilir ve içeriği değiştirilebilir. FAT kütüphanesinin tüm özelliklerini kullanabilmek için uygun MSP430 denetleyici ile FatFs kütüphanesini kullanmak gerekmektedir.

Petit FatFs Kütüphanesinin MSP430 Denetleyicilerle Kullanımı

Petit FatFs kütüphanesi kendi içerisinde bir çok dosyadan oluşmaktadır. Bu yüzden dolayı modülde kütüphane kodlarına yer verilmemiştir. Petit FatFs kütüphanesi genel bir kütüphane olduğu için MSP430 denetleyiciler için bazı alt seviye fonksiyonların yazılması gerekmektedir.

SD kartların SPI ile haberleşebildiğinden bahsetmiştik. MSP430 denetleyicilerde bulunan USCI biriminin SPI haberleşmeyi desteklemesinden dolayı SD kart ile haberleşmede bu birimden faydalanılır. Sonuç olarak USCI birimini kullanarak SD kart ile SPI üzerinden haberleşmek için gerekli alt seviye fonksiyonların hazırlanması gerekmektedir. Biz uygulamalarımızda MSP430 denetleyiciler için uyarlanmış hazır Petit FatFs kütüphanesinden faydalanacağız. MSP430 geliştirme kartımız üzerinde bulunan SD kart soketi MSP430 denetleyicimizin USCI_B0 birimi ile SPI üzerinden haberleşecek şekilde tasarlanmıştır.



Şekil 4: MSP430 Geliştirme Kiti SD Kart Uygulama Şeması

Şekil 4'te MSP430 geliştirme kiti üzerinde bulunan SD kart biriminin devre şeması görülmektedir. Görüldüğü gibi SD kart ile MSP430 haberleşmesi için 4 adet I/O pini yeterlidir. 4 adet I/O pini ile SPI üzerinden SD kart ile iletişim kurup uygulamalarımızda kullanabiliriz.

Uygulama 10.1 SD Kart Kullanımı

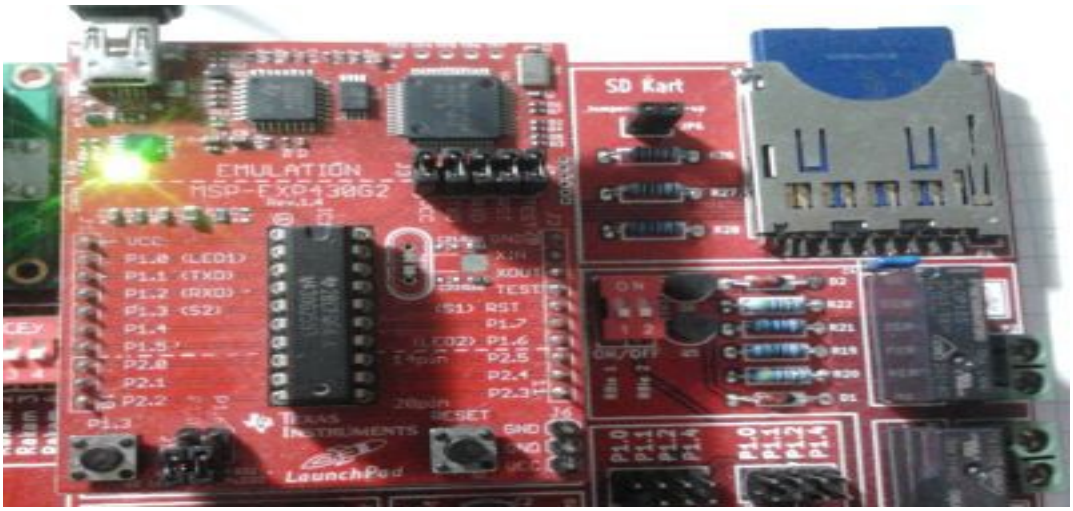
Bu uygulamamızda SD kartı MSP430 denetleyici ile haberleştirip çalışmasını göreceğiz. Uygulamamız SD kart içerisinde bulunan “deneme.txt” isimli metin belgesinden 64 karakter veri okuyacak. Okuduğu karakterler küçük harf ile yazılmışsa büyük harfe çevirip tekrar geri yazacak. Dosyanın boyutunda bir değişiklik olmayacağı için bu şekilde içeriğini değiştirebiliriz. Uygulama kodları aşağıdaki gibidir.

```
#include <msp430.h>       // MSP430 başlık dosyası

// SD kart kullanımı ile ilgili kütüphane dosyaları
#include <stdint.h>
#include "pff2a/src/diskio.h"
#include "pff2a/src/pff.h"
#include "drivers/spi.h"
////////////////////////////////////////////////
// Kullanılan fonksiyon prototipleri
void Yanlis(void);
void Dogru (void);
void Bekle(unsigned int);
// Dosya sisteminde kullanılan değişkenler
        FATFS fatfs;    // File system object
        WORD  br,bw;
        BYTE buff[64];
////////////////////////////////////////////
        unsigned char bSayac;    // Sayaç değişkeni

void main (void)
{
        WDTCTL  =  WDTPW + WDTHOLD;             // Watchdog timeri durdur
        WDTCTL  = WDTPW | WDTHOLD;              // Watchdog timeri durdur.
        BCSCTL1 = CALBC1_16MHZ;                 // Dahili osilatörü 16MHz'e ayarla.
        DCOCTL  = CALDCO_16MHZ;                 // Dahili osilatörü 16MHz'e ayarla.
        P1DIR       &= ~BIT0;                   // Port1.0 giriş
        P1SEL       &= ~BIT0;                   // Port1.0 I/O olarak kullanılacak
        P1SEL2      &= ~BIT0;                   // Port1.0 I/O olarak kullanılacak
        P1REN       |= BIT0;                    // Port1.0 Pull-up/down direnci aktif
        P1OUT       |= BIT0;                    // Port1.0 Pull-up özelliği aktif
        P2DIR|= BIT1 + BIT2;                    // Port2.1-2 çıkış
        P2SEL       &= ~(BIT1 + BIT2);          // Port2.1-2 I/O olarak kullanılacak
        P2SEL2      &= ~(BIT1 + BIT2);          // Port2.1-2 I/O olarak kullanılacak
        P2OUT       &= ~(BIT1 + BIT2);          // Port2.1-2 çıkışlarını sıfırla
        spi_initialize();                       // USCI_B0 birimini SPI için hazırla

        while(1) {                              // Sonsuz döngü
        while(P1IN & BIT0);                     // Butona(P1.0) basılmasını bekle
        _delay_cycles(16000);                   // Buton arkını önlemek için bir süre bekle
        while(!(P1IN & BIT0));                  // Butonun(P1.0) bırakılmasını bekle
        if(pf_mount(&fatfs)==RES_OK){           // Kartı aç
        if(pf_open("deneme.txt")==RES_OK){      // deneme.txt doyasını aç
        if(pf_read(buff,64, &br)==RES_OK){      // Dosyadan 64 karakter veri oku
        for (bSayac=0; bSayac<=63; bSayac++)    // Okunan karakterleri kontrol et
        {
```

```
        if( buff[bSayac] >= 'a' && buff[bSayac] <= 'z')   // Karakter küçük harf mi?
        buff[bSayac] -= 0x20;                             // Evet ise büyük harf yap.
        }
        if(pf_write(buff, 64, &bw)==RES_OK){   // Okunan karakterleri geri yaz
        if(pf_write(0, 0, &bw)==RES_OK){       // Dosyayı kapat
        Dogru();                               // Doğru uyarısı ver
        } else Yanlis();                       // Dosya kapanmadıysa yanlış uyarısı ver
        } else Yanlis();                       // Karakterler yazılmadıysa yanlış uyarısı ver
        } else Yanlis();                       // Karakterler okunmadıysa yanlış uyarısı ver
        } else Yanlis();                       // Dosya açılmadıysa yanlış uyarısı ver
        } else Yanlis();                       // Kart açılmadıysa yada yoksa yanlış uyarısı ver
        }                                      // Sonsuz döngü bitimi
}                                              // Ana program sonu

// Port2.1 pinine bağlı LED ile uyarı veren fonksiyon
void Dogru(void){
        P2OUT |= BIT1;
        Bekle(1000);
        P2OUT &= ~BIT1;
        Bekle(500);
        P2OUT |= BIT1;
        Bekle(500);
        P2OUT &= ~BIT1;
        Bekle(500);
}

// Port2.2 pinine bağlı LED ile uyarı veren fonksiyon
void Yanlis(void){
        P2OUT |= BIT2;
        Bekle(500);
        P2OUT &= ~BIT2;
        Bekle(500);
        P2OUT |= BIT2;
        Bekle(500);
        P2OUT &= ~BIT2;
        Bekle(500);
}

// Genel amaçlı bekleme fonksiyonu
void Bekle (unsigned int Bekle){
        unsigned int i;
        for(i=0;i<Bekle;i++)
        _delay_cycles(16000);
}
```

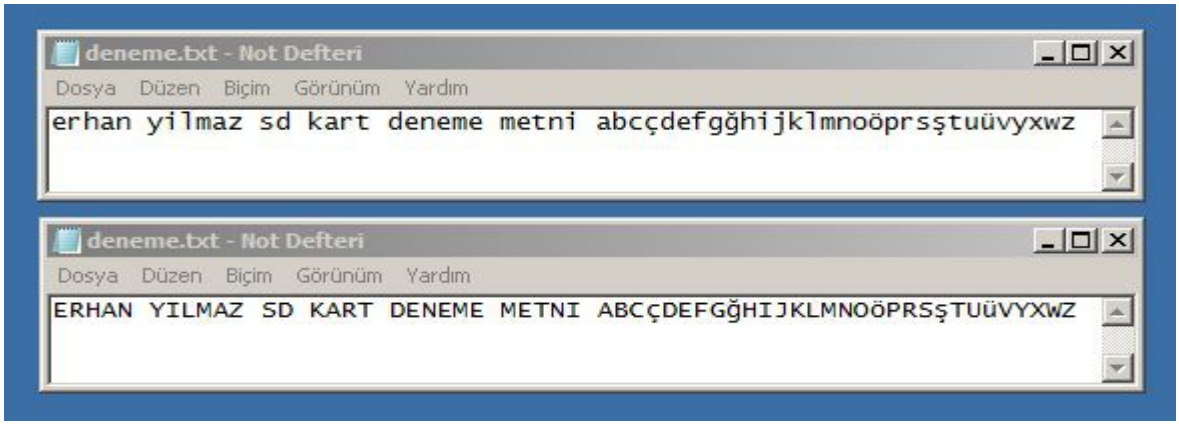
Kodları inceleyecek olursak, diğer uygulamalardan farklı olarak bu uygulamamızda dahili osilatör 16 MHz yani azami hızda çalıştırılmıştır. Bunun nedeni SD kart ile yüksek hızda haberleşme yapabilmek içindir.
Genel ayarlar yapıldıktan sonra SPI ayarları yapılarak program sonsuz döngüye girer. Sonsuz döngüde Port1.0 pinine bağlı butona basılması beklenir. Butona basıldıktan sonra bırakılması beklenir. Buton basılıp bırakıldıktan sonra, önce SD kart içinde bulunan hafıza birimi açılır. Sonrasında kart içerisinde deneme.txt dosyası var ise açılır. Dosya açıldıktan

sonra içinden ilk 64 karakterlik veri okunur. Okunan karakterlerin küçük karakter olup olmadığı kontrol edilerek, küçük olan karakterler büyük karaktere çevirilir. Çevirme işlemi bitince değerler tekrardan geri yazılır. Sonrasında dosya kapatılarak işlem sonlandırılır. Eğer tüm işlemler sorunsuz icra edilirse Port2.1 pinine bağlı LED doğru uyarısı verir. Eğer işlemlerde herhangi bir sorun olursa, örneğin SD kart takılı değil yada içerisinde deneme.txt isimli dosya yoksa Port2.2 pinine bağlı LED yanlış uyarısı verir. Bu şekilde uygulama sürekli çalışmasına devam eder.



Şekil 5: Uygulama 10.1 Çalışma Görüntüsü

Şekil 5'te görüldüğü gibi uygulamayı deneyebilmek için SD kartın P4 SD kart soketine yerleştirilmesi gerekir. Uygulamada 2G byte'lık SD kart kullanılmıştır. SD kart bulamaz iseniz mikro yada mini SD kart ile SD kart dönüştürücü adaptör kullanabilirsiniz. Uygulamayı çalıştırmadan önce geliştirme kartı ayarlarının yapılması gerekir. Bunun için jp5 atlaması on, jp1 atlaması gnd, sw4 üzerinde bulunan 1 numaralı anahtar on ve sw15 üzerinde bulunan 2 ve 3 numaralı anahtarlar on konumuna getirilmelidir. Ayrıca SD kart soketinin yanında bulunan jp6 atlamasıda takılmalıdır. Son olarak sorun çıkmaması için geliştirme kartı üzerinde bulunan diğer atlama ve anahtarların kapalı konuma getirilerek kullanılmayan birimlerin kapatılmasında fayda var. Gerekli ayarlamaları yaptıktan sonra kodları derleyip geliştirme kartımıza yükledikten sonra herhangi bir sorun yoksa uygulamamız çalışacaktır.



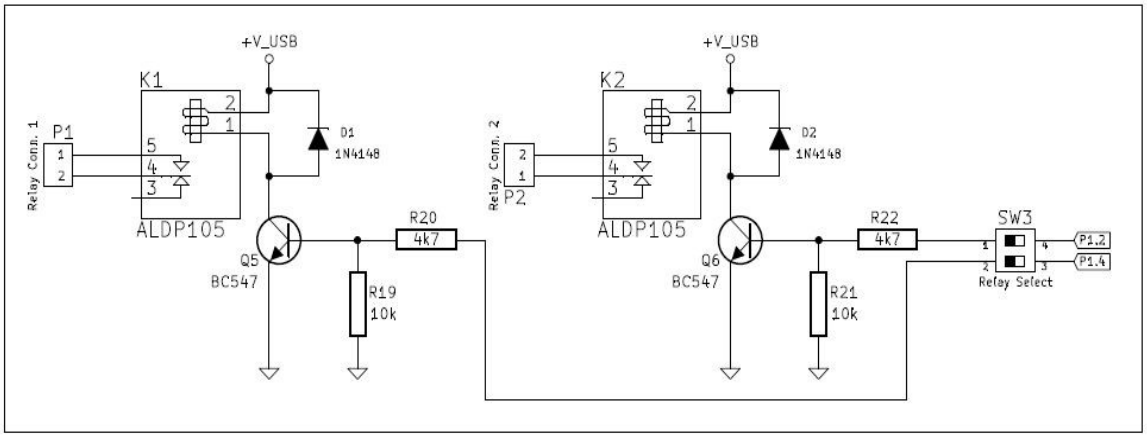
Şekil 6: Uygulama 10.1 Çalışma Görüntüsü

Şekil 6'da görüldüğü gibi üstteki metin belgesi deneme.txt dosyasının orijinal halidir. Dosyayı SD karta yükleyip SD kartı geliştirme kartına takıp Port1.0 butonuna basıp bırakınca eğer bir hata yok ise metin içerisindeki karakterler büyük karakterlere çevirilir. Görüldüğü gibi Türkçe karakterlerde her hangi bir değişiklik olmamıştır. Bunun nedeni Türkçe karakterlerin ASCII karakter tablosunda olmayışıdır.

Uygulama 10.2 SD Kart ile Şifreli Kilit

Bu uygulamamızda değişik bir şifreli kilit yapacağız. SD kart içerisinde sifre.txt isimli metin belgesi oluşturacağız. Metin belgesi içerisine 4 karakterlik bir şifre yazacağız. (örneğin '1234') Uygulamada SD kart içerisinde bulunan sifre.txt isimli dosyanın ilk 4 karakterini okuyup program içerisinde tanımladığımız şifre ile karşılaştıracağız. Şifre doğru ise kart üzerinde bulunan röleleri çektireceğiz. Böylece SD kart ile çalışan şifreli kilit uygulaması yapıp rölelerin kullanımını da değineceğiz.

MSP430 geliştirme kartı üzerinde 2 adet röle bulunmaktadır. Röleler bilindiği gibi küçük akımlarla büyük güçteki cihazları kontol etmeye yarar. Kart üzerindeki röleler tampon devreler üzerinden MSP430 denetleyicinin I/O pinlerine bağlıdır. Bu sayede LED yakıp söndürür gibi rölerleri açıp kaptabiliriz. Rölerleri direk olarak denetleyiciler ile sürmek sakıncalıdır. Aşırı akım çekip denetleyici portlarına zarar verebilirler. Bu yüzden çoğunlukla denetleyiciler ile röle kullanırken basit tampon devreleri kullanılır.



Şekil 7: MSP430 Geliştirme Kartı Röle Uygulama Devresi

Şekil 7'de MSP430 geliştirme kartı üzerinde bulunan rölelerin bağlantı şeması görülmektedir. Kart üzerinde 2 adet ALDP105 isimli röle bulunmaktadır. Bu röleler 5V DC gerilim ile çalışıp 5A-250V AC çalışma akım ve gerilimine sahip cihazların aç/kapa kontrolünü yapabilirler. Bu sayede MSP430 denetleyici ile yüksek güç ile çalışan lamba, tv, ısıtıcı gibi cihazların kontrolünü yapabilirsiniz. Röleler denetleyici portuna transistörlü basit tampon devreleri ile bağlıdırlar. Bu sayede direk olarak port çıkışlarını aktif/pasif ederek rölelerin kontrolünü yapabiliriz.

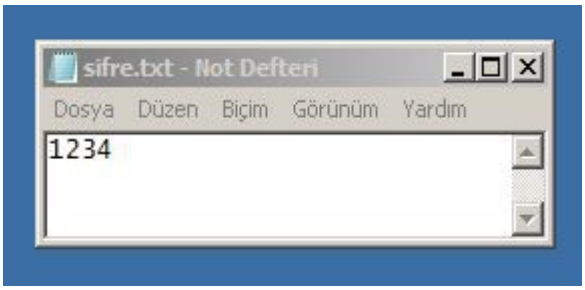
Uygulama 10.2'de bu röleler kullanılacaktır. Röle çıkışlarına bir elektirkli kapı kilidi takıp şifrenin doğru olması durumunda kapının açılması sağlanabilir. Uygulama kodları aşağıdaki gibidir.

```
#include <msp430.h>       // MSP430 başlık dosyası

// SD kart kullanımı ile ilgili kütüphane dosyaları
#include <stdint.h>
#include "pff2a/src/diskio.h"
#include "pff2a/src/pff.h"
#include "drivers/spi.h"
//////////////////////////////////////////////////////
// Kullanılan fonksiyon prototipleri
```

```
void Yanlis(void);
void Dogru (void);
void Bekle(unsigned int);
// Dosya sisteminde kullanılan değişkenler
        FATFS fatfs;    // File system object
        WORD  br,bw;
        BYTE buff[4];
//////////////////////////////////////////
        unsigned char bSayac;                   // Sayaç değişkeni
        unsigned char bSifre[4]={'1','2','3','4'};  // Şifre "1234"

void main (void)
{
        WDTCTL  =  WDTPW + WDTHOLD;     // Watchdog timeri durdur
        WDTCTL  = WDTPW | WDTHOLD;      // Watchdog timeri durdur.
        BCSCTL1 = CALBC1_16MHZ;         // Dahili osilatörü 16MHz'e ayarla.
    DCOCTL = CALDCO_16MHZ;              // Dahili osilatörü 16MHz'e ayarla.
        P1DIR       &= ~BIT0;           // Port1.0 giriş
        P1DIR|= BIT2 + BIT4;            // Rölelerin bağlı olduğu pinleri çıkış yap
        P1SEL       &= ~(BIT0 + BIT2 + BIT4); // Port1.0,2,4 I/O olarak kullanılacak
        P1SEL2      &= ~(BIT0 + BIT2 + BIT4); // Port1.0,2,4 I/O olarak kullanılacak
        P1REN       |= BIT0;            // Port1.0 Pull-up/down direnci aktif
        P1OUT       |= BIT0;            // Port1.0 Pull-up özelliği aktif
        P1OUT       &= ~(BIT2 + BIT4);  // Röleleri başlangıçta kapat
        P2DIR|= BIT1 + BIT2;            // Port2.1-2 çıkış
        P2SEL       &= ~(BIT1 + BIT2);  // Port2.1-2 I/O olarak kullanılacak
        P2SEL2      &= ~(BIT1 + BIT2);  // Port2.1-2 I/O olarak kullanılacak
        P2OUT       &= ~(BIT1 + BIT2);  // Port2.1-2 çıkışlarını sıfırla
        spi_initialize();               // USCI_B0 birimini SPI için hazırla

        while(1) {                      // Sonsuz döngü
        while(P1IN & BIT0);             // Butona(P1.0) basılmasını bekle
        _delay_cycles(16000);           // Buton arkını önlemek için bir süre bekle
        while(!(P1IN & BIT0));          // Butonun(P1.0) bırakılmasını bekle
        if(pf_mount(&fatfs)==RES_OK){   // Kartı aç
        if(pf_open("sifre.txt")==RES_OK){   // sifre.txt doyasını aç
        if(pf_read(buff,4, &br)==RES_OK){   // Dosyadan 4 karakter şifreyi oku
        if(bSifre[0]==buff[0])          // Şifrenin 1. hanesini kontrol et
        if(bSifre[1]==buff[1])          // Şifrenin 2. hanesini kontrol et
        if(bSifre[2]==buff[2])          // Şifrenin 3. hanesini kontrol et
        if(bSifre[3]==buff[3])          // Şifrenin 4. hanesini kontrol et
        Dogru();            // Şifre doğru ise doğru uyarısı ver ve röleleri çektir.
        else                // Yanlış ise?
        Yanlis();           // Yanlış uyarısı ver
        pf_write(0, 0, &bw);// Dosyayı kapat
        } else Yanlis();    // Karakterler okunmadıysa yanlış uyarısı ver
        } else Yanlis();    // Dosya açılmadıysa yanlış uyarısı ver
        } else Yanlis();    // Kart açılmadıysa yada yoksa yanlış uyarısı ver
        }                   // Sonsuz döngü bitimi
}                           // Ana program sonu
```

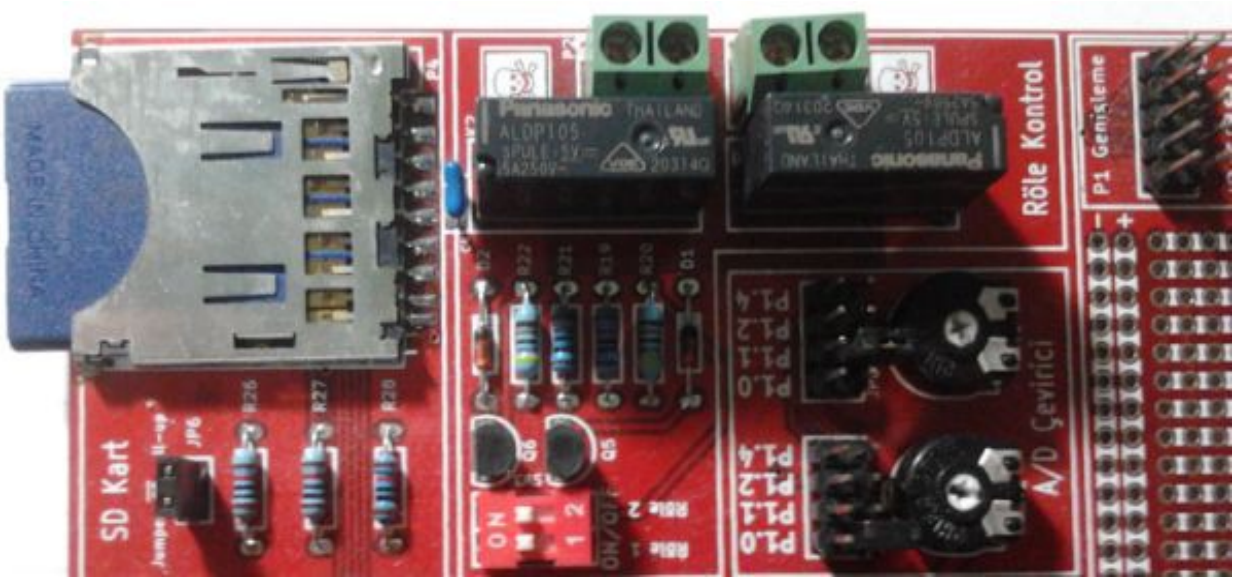
```
// Doğru uyarısı verip röleleri çektiren fonksiyon
void Dogru(void){
        P2OUT |= BIT1;
        P1OUT |= BIT2 + BIT4;           // Röleleri çektir
        Bekle(1000);                    // Bir süre bekle
        P2OUT &= ~BIT1;
        P1OUT &= ~(BIT2 + BIT4);        // Röleleri bırak
        Bekle(500);
}

// Port2.2 pinine bağlı LED ile uyarı veren fonksiyon
void Yanlis(void){
        P2OUT |= BIT2;
        Bekle(500);
        P2OUT &= ~BIT2;
        Bekle(500);
        P2OUT |= BIT2;
        Bekle(500);
        P2OUT &= ~BIT2;
        Bekle(500);
}

// Genel amaçlı bekleme fonksiyonu
void Bekle (unsigned int Bekle){
        unsigned int i;
        for(i=0;i<Bekle;i++)
        _delay_cycles(16000);
}
```

Uygulama kodları uygulama 10.1 ile büyük oranda benzerdir. Aynı şekilde başlagnıç ayarları yapıldıktan sonra program sonsuz döngüye girer. Başlangıç ayarlarında rölerin bağlı olduğu Port1.2,4 pinleri çıkış olarak ayarlanır ve rölerler başlangıçta kapalı tutulur. Sonsuz döngüde Port1.0 pinine bağlı butona basılıp bırakılması beklenir. Butona basıp bıraktıktan sonra uygulama SD kart içerisinde bulunan sifre.txt doyasını açılıp içerisinde bulunan ilk 4 byte karakteri okunur. Okunan bu değerler programda belirlenmiş olan şifre ile karşılaştırılır. Eğer şifre doğru ise Port2.1 pinine bağlı LED doğru uyarısı veri ve röleler bir süre çektirilip bırakılır. Eğer şifre yanlış ise yada SD kart ile ilgili bir sorun var ise Port2.2 pinine bağlı LED ile yanlış uyarısı verilir. Bu şekilde uygulama sürekli çalışmasına devam eder.



Şekil 8: sifre.txt Metin Belgesi

Şekil 8'de görüldüğü gibi sifre.txt isimli metin belgesi hazırlanıp SD karta yüklenir. Uygulamada örnek şifre '1234' karakterleridir.



Şekil 9: Uygulama 10.2 Çalışma Görüntüsü

Uygulama kodlarını geliştirme kartına yüklemeden önce kart ayarlarının yapılması gerekir. Bunun için jp5 atlaması on, jp1 atlaması gnd, sw3 anahtarları on, sw4 üzerinde bulunan 1 numaralı anahtar on ve sw15 üzerinde bulunan 2 ve 3 numaralı anahtarlar on konumuna getirilmelidir. Ayrıca SD kart soketinin yanında bulunan jp6 atlamasıda takılmalıdır. Son olarak sorun çıkmaması için geliştirme kartı üzerinde bulunan diğer atlama ve anahtarların kapalı konuma getirilerek kullanılmayan birimlerin kapatılmasında fayda var.
Geliştirme kartı ayarlarını yaptıktan sonra uygulamala kodlarını derleyip yükleyince herhangi bir sorun yoksa uygulama çalışacaktır. İçerisinde şifrenin bulunduğu sifre.txt isimli metin belgesini SD karta yükledikten sonra SD kartı geliştirme kartına takalım. Sonrasında geliştirme kartı üzerinde bulunan Port1.0 pinine bağlı sw14 butonuna basıp bırakalım. Eğer hata yok ise ve şifre doğru ise port2.1 pinine bağlı LED doğru uyarısı verip röleler bir bir süre çekilip bırakılacaktır. Şifre yanlış ise yada SD kart ile ilgili bir sorun olursa uygulama Port2.2 pinine bağlı LED ile yanlış uyarısı verecektir. Bu sayede röleye bağlı kapı kiliti çalıştırılıp kapı kontolü sağlanabilir.

Bu modül kapsamında SD kartlardan bahsedilip MSP430 denetleyiciler ile uygulamaları yapılmıştır. Ayrıca geliştirme kartı üzerinde bulunan rölerlerin uygulaması yapılmıştır. SD kartlar yüsek depolama alanı, pratik ve hızlı olması gibi özelliklerinden dolayı yoğun olarak kullanılmaktadır. SD kartlar MSP430 denetleyiciler ile kullanılıp daha işlevsel ve pratik uygulamalar geliştirilebilir.