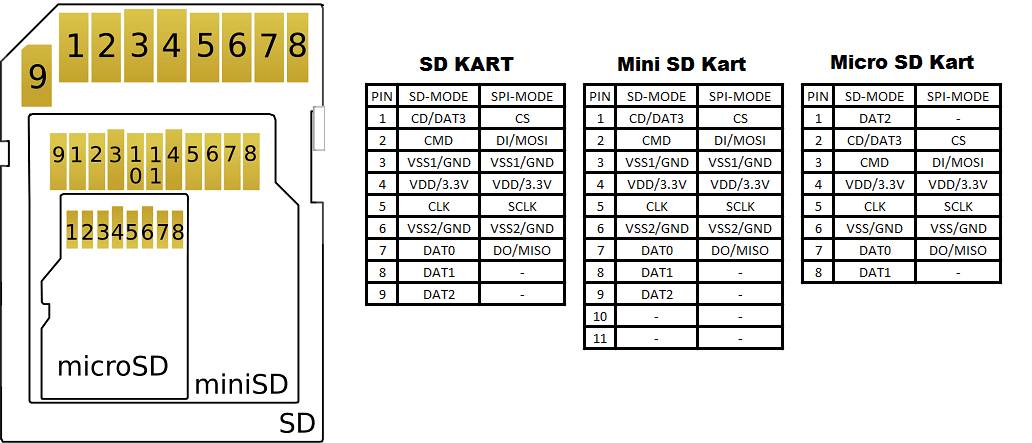
**SD KART NEDİR?**

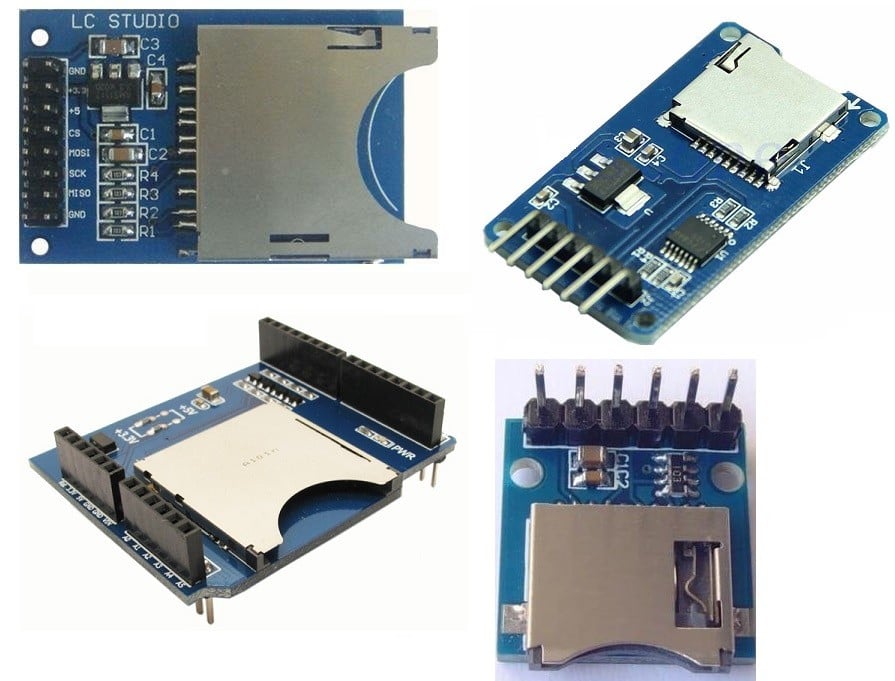
SD Memory Card yani “Secure Digital Memory Card” kısaltması olan SD kartlar bir veri depolama aygıtıdır. İlk kez SanDisk firması tarafından 2001 yılında MMC kart teknolojisinin geliştirilmesi üzerine ortaya çıkmıştır. SD kartlar iki farklı iletişim protokolüne sahiptir; SD modu ve SPI modu.

Arduino IDE içindeki dahili SD kart kütüphanesi SPI modunda SD kart ile iletişim kurmaya izin verdiğinden dolayı SPI iletişim protokolü ile SD kartımızla haberleşme sağlamış oluyoruz. SD kartlar 2.0V – 3.6V arasında besleme ile çalışmaktadırlar. Bundan dolayı SD kartları 5V ile çalışan Arduino’lar ile doğrudan bağlayamayız. Biçim yani boyut olarak 3 tipte SD kart bulunmaktadır ve bunlar arasında bacak sıralamaları arasında farklılıklar dışında SPI modu için gerekli başka farklılıklar bulunmamaktadır. Önce SD kartların bacak isimleri ve numaralarını inceleyelim. Aşağıda SD kart bacak isimlendirmeleri görülmektedir.

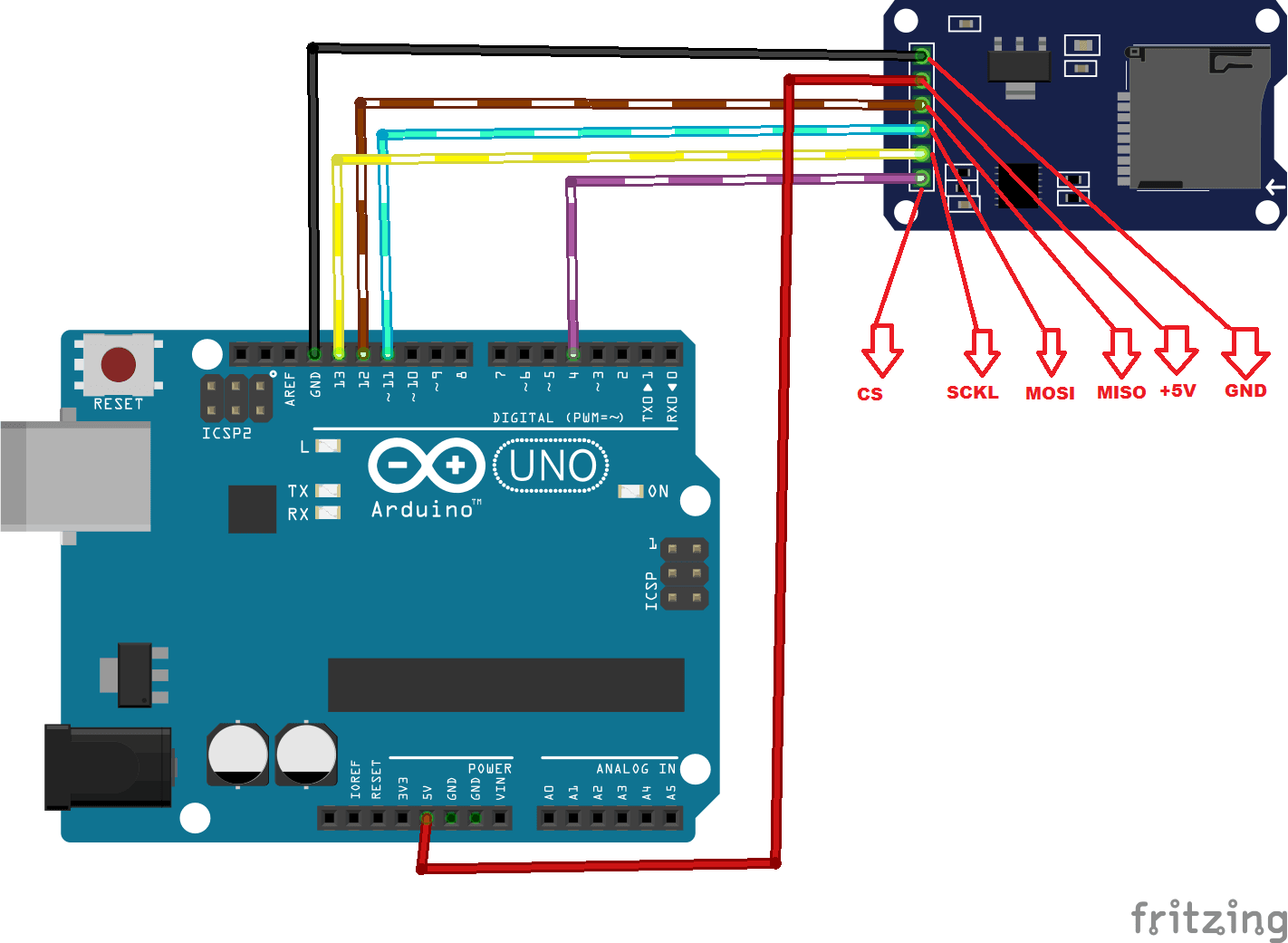


Dikkat ederseniz normal mini veya micro SD kart olsun hepsinde aynı bacaklar var ancak fiziksel olarak sıralanışında farklılık bulunmaktadır. Bizim için önemli olan SPI modu için kullanılan bacaklarıdır diğer bacaklar boşta kalabilir. Bağlantı Arduino ile SD kart arasında MISO-MISO , MOSI-MOSI , SCLK-SCLK , SS(CS)-CS şeklinde olup VDD için 3.3V ve VSS için GND bağlantısının yapılmasıdır. Ancak MOSI, MISO, SCLK ve CS hatlarının 10kOhm ile 3.3V’a Pull-Up yapılması gerekmektedir.

Bu işlemi bu şekilde yapan veya “logic-level converter” çipli hazır SD kart modülleri veya shield’lar piyasada mevcuttur. İster çipli olsun ister dirençli olsun sonuç olarak bizim 3.3V beslemeye ihtiyacımız var. Bazı modüller üzerinde AMS1117-3.3 3.3V regülatörler bulunurken bazılarında bulunmamaktadır. Çok önemli değil nasıl olsa 3.3V çıkışı neredeyse bütün Arduino kartlarında mevcut. Modüllerin hepsine de aynı bağlantılar bulunmaktadır bu yüzden şu modül bu modül nasıl kullanılır farkı var mı demeyin yok. Aşağıda bu modüllere ait görseller görülmektedir.



Aşağıda Arduino ile SD kart modülün minimum bağlantı şeması görülmektedir. fritzing de bende sadece bu SD kart modülü vardı sizdeki modülün bacak sıralaması farklı olabilir ona göre bağlayınız.



Şimdi gelelim esas kısma. Donanımsal olarak her şeyi hazırladık şimdi gelin Arduino ile SD kartımızı nasıl kullanacağımızı görmeye. Bunun için Arduino’nun kendi SD kart kütüphanesini kullanacağız.

**SD KART FONKSİYONLARI VE KULLANIMI**

***!!! Her zaman en fazla bir adet dosya açılabilir. Bir diğer dosyayı açabilmek için önce açık olan dosya kapatılmalıdır ardından başka bir dosya açabilirsiniz !!!***  
***!!! Buna dikkat ediniz çünkü SD kart kullanırken yapabileceğiniz en muhtemel hata bundan kaynaklı olacaktır !!!***

**SD KART KÜTÜPHANESİNİ DAHİL ETME VE DOSYA NESNESİ OLUŞTURMA﻿**

Öncelikle Dosya Nesnesine ihtiyacımız var.

* **File dosya;**

Bu şekilde “dosya” adında bir dosya nesnesi tanımladık. Bu nesne ile SD karttan açacağımız dosyaya diğer fonksiyonlar ile veri yazabilir, okuyabilir veya dosyayı kapatabiliriz. Adını dosya koydum siz kendinize göre bir isim verebilirsiniz, mesela;

* **File benimDosyam;**

**SD KARTI BAŞLATMA**  
Bundan sonraki adım SD kartımızı hazır hale getirmek olacak.

* **SD.begin(parametre);**

Bu fonksiyon ile SPI haberleşme başlatılır ve hemen ardından SD kart takılı mı değil mi kontrol edilir.

Parametre olarak girilen rakam bizim dijital pinlerimizden birini ifade ediyor ve SD kartımızın CS bacağı yani SD kartı SPI hattında seçmeye yarayan Arduino pini olmaktadır. Mesela 4 yazarsanız 4 numaralı digital pini SD kart modülünün CS bacağın bağlamalısınız. Eğer SD kart takılı ise ve bir arıza yok ise bu fonksiyon 1 sonucunu üretiyor yani HIGH, SD kart takılı değil veya arızalı ise bu fonksiyon 0 sonucunu geri döndürür yani LOW.

Artık SD kartımızda dosya açabilir içine bilgi yazabilir içini okuyabilir dosyayı kapatabilir kısaca herşeyi yapabiliriz.

**DOSYALARI OKUMAK VEYA YAZMAK İÇİN AÇMA VE DOSYA OLUŞTURMA﻿**

Bir dosyayı okuyabilmek veya içine veri yazabilmek için öncelikle bir dosyayı açmamız gerekmektedir.

* **dosya = SD.open(“deneme.txt”,FILE\_WRITE);**

İsmi belirtilen var olan dosyayı açan veya belirtilen isimde dosyanın olmaması durumunda belirtilen isimde bir dosya oluşturan daha sonra bu dosyayı açıp içine veri yazmaya hazır hale getiren kod satırı.

* **dosya = SD.open(“deneme.txt”,FILE\_READ);**

İsmi belirtilen var olan dosyayı açan daha sonra bu dosyayı açıp içindeki verileri okumaya hazır hale getiren kod satırı.

“deneme.txt” adlı dosya ismi keyfidir, orada “test.txt” isminde bir dosya adı da olabilirdi. Yalnız dosya isimlendirmeleri ile ilgili bir kısıtlama mevcut, en fazla 8 karaktere kadar isimler verebilirsiniz.

Her iki durumda da SD.open(…); fonksiyonu oluşturduğumuz “dosya” nesnesine eşitlenmektedir! Bunun kesinlikle yapılması gerekmektedir.  
Aksi halde açılan dosyalara yazma veya okuma işlemini gerçekleştiremeyiz.

**SD KARTA VERİ YAZMAK**

Şimdi de sıra geldi açtığımız bu dosyaya veri kaydetmeye.

* **dosya.print(…);** ve   **dosya.println(…);**

Buradaki “dosya.print();” ve “dosya.println();” aynı “Serial.print();” ve “Serial.println();” şeklinde görev yapıyor. Tek fark verileri Serial monitöre değil, SD kart içerisinde daha önceki kod satırı ile açtığımız dosyaya yazıyor olmasıdır.

Başlarındaki “dosya” ibaresi en başta “File dosya;” şeklinde tanımladığımız dosya adlı nesnemizdir.

**SD KARTTA AÇILAN DOSYAYI KAPATMAK**

Bu sefer de açtığımız ve içerisine veri kaydettiğimiz verileri o dosyadan okumak istedik diyelim. Ancak en baştaki uyarıya dikkat çekmek isterim, eğer içerisine veri kaydetmek üzere açılmış bir dosya varsa veya okunmak istenen başka bir dosya ise önce açık olan mevcut dosyanın kapatılması lazım.

* **dosya.close();**

Bu fonksiyon ile o anda en son açılmış olan dosya kapatılır.

**SD KARTTAN VERİ OKUMA VE OKUMA YÖNTEMLERİ**

Şimdi var olan veya kendimizin açıp içine bir şeyler kaydettikten sonra kapattığımız bir dosyayı açıp içindeki verileri okuyalım. Bunun için dosyayı açarken kullanacağımız kod satırındaki parametre ” FILE\_READ ” şeklinde daha önce bahsettiğim gibi ” dosya = SD.open(“deneme.txt”,FILE\_READ); ” olmalıdır.

Şimdi tüm herşey karman çorman olacak. Eğer ASCII tablosu ve RS-232 ile hiç uğraşmadıysanız ya da Serial monitörden girilen rakamları Arduino ile nasıl hesaplama için kullanırım şeklinde uğraşlarınız olmadıysa şimdi anlatacaklarım sizlere yabancı gelebilir.

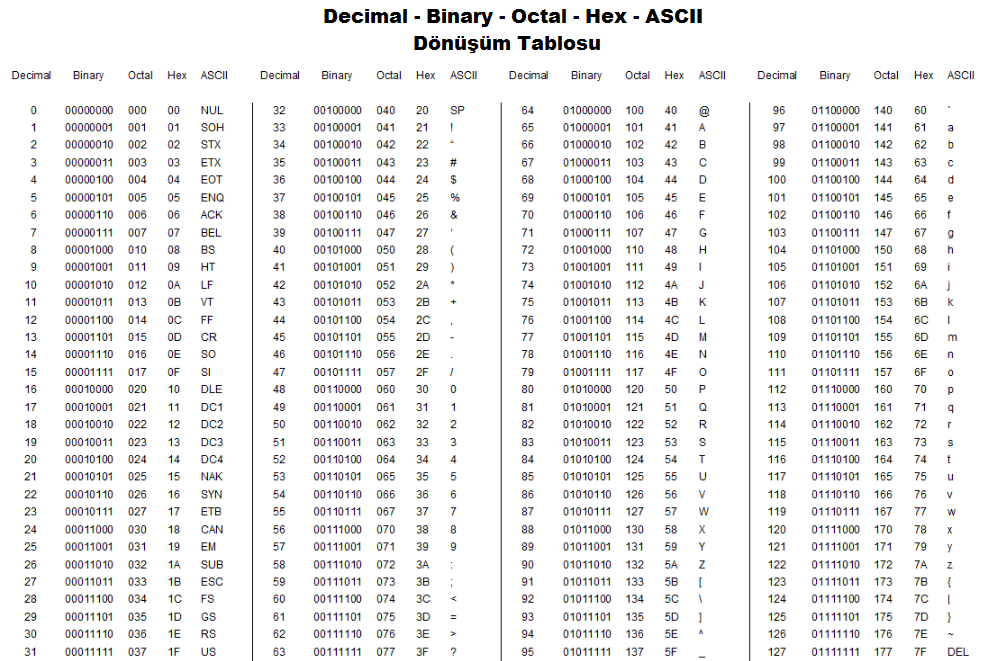
Klavyeden bastığınız ve ekranda gördüğünüz her karakter bilgisayarda bizim gördüğümüz şekilde yorumlanmamaktadır. Mesela A harfi ile a harfi. İki harfte aynı ama biri küçük ve biri büyük bu ayrımı yapan bilgisayarın harflerin boyutlarını ayırt etmesiyle değil ASCII kodlarına göre bunu bilmesidir.

Her karakterin bir sekizlik, onluk, onaltılık, HEX ve Binary sayı değeri vardır. Örneğin A harfine karşılık gelen onluk sayı 65’tir. Eğer klavyenizin Alt tuşuna basılı tutup 65 rakamı yazarsanız A harfini yazmış olursunuz. Yani siz A harfine ait tuşa bastığınızda Alt+65 görevini yerine getirmektedir bilgisayar.

dosya.print(“LRT”); veya Serial.print(“LRT”); ile SD karta veya Serial monitöre LRT yazısını gönderirken ASCII onluk karşılıklarını her harf için arka arkaya yollamaktadır.

dosya.println(13.09); veya Serial.print(13.09); ile SD kart veya Serial monitöre 13.09 ondalıklı sayıyı göndermektedir. Bu gönderilen bir karakter değildir, rakamdır! LRT örneğinde gönderilenler karakterdi! Ancak gönderirken bunları kendisi yine karaktere çevirmekte ve ardından her karaktere ait ASCII onluk sayı karşılığını arka arkaya yollamaktadır.

Rakam ile karakter arasındaki farkı hep şöyle hatırlatırım kendime. 1 sayısı binary olarak 00000001’dir, ancak 1 karakteri binary olarak 00110001’dir. Eğer sizde benim gibi biraz sıyırıp ASCII tablosunu ezbere bilmiyorsanız aşağıdaki ASCII tablosuna bakabilirsiniz. Decimal onluk, Binary birlik, Octal sekizlik ve HEX(hexadecimal) onaltılık tabanda sayılardır. Karakterler ASCII olarak tanımlanırlar.



Ben bunu anlatıyorum çünkü bir şeyleri yollarken yani veri yazı gibi bilgileri karşı tarafa Serial.print ile gönderirken alıcı tarafından hiç hangi türde veri gönderiyoruz diye düşünmediğimizden anlatıyorum. SD kart okumayla bunların ne ilgisi olduğunu az da olsa anlamış olmalısınız. Bizler projelerimizde SD kartları genelde datalogger ( veri kayıt edici ) olarak kullanmaktayız. Veri olarak kaydettiğimiz şeylerin bir çoğuda rakamlardan oluşmaktadır. Harfler ise bizlere açıklama sağlayabilir ancak hesaplanabilir anlamlı bilgiler rakamlardır.

Biz SD karta bir rakam kaydettiğimiz zaman bunu ASCII yani karakter olarak kaydetmektedir. Eğer bu kaydedilen bilgiyle yani karaktere dönüşmüş rakam bilgimizle matematiksel işlemleri tekrar yapmak kullanamayız! Önce bu karaktere dönüştürülmüş olan bu bilgiyi tekrar rakam dönüştürmemiz gerekmektedir.  
Ancak eğer yok ben SD karta kayıt ettiğim bilgileri bilgisayarda işleme tabi tutacağım diyorsanız buna bir şey diyemem bu da bir yöntemdir ki benim de tavsiye ettiğim yol da bu olurdu. Ancak kayıt edilen bilgilerin bazen yine Arduino veya mikrodenetleyiciler ile işlenmesi gerekebilir.

Şimdi gelin SD karttan veriler nasıl okunur bunu görelim. Tabi hatırlamak için tekrar yazıyorum nesne adımız “dosya” bunu unutmayalım.

* **dosya.read();**

Bu fonksiyon ile SD kartta okumak için açtığımız dosyanın en başındaki ilk karakterini okumamıza yaramaktadır. Dosya kapatılana kadar bu fonksiyon ne kadar arka arkaya çağrılırsa çağrılsın her seferinde bir sonraki karakteri okumaktadır. Mesela LM35 ile yarım saate bir, on saat boyunca toplamda 20 adet farklı sıcaklık ölçümünü SD karta kaydettik ve bunları şimdi okuyacağız. Her satırda her yarım saate bir kaydedilen bir bilgi bulunmakta mesela ilk ölçümde kayıt olan sıcaklık değeri “25.24” ilk ölçüm ilk kayıt anında dosyanın en başına kayıt olacağından dosya.read(); ile bu ilk bilgiyi komple okumamız için 4 rakam ve bir nokta karakteri için toplamda 5 kere dosya.read(); fonksiyonunu arka arkaya yazmamız gerekmektedir.

Ancak bazen o kadar çok veri kaydolmuş olabilir ki nerede biteceğini bilemeyiz ve sonuna gelsek bile okumanın bunu dosya.read(); fonksiyonu algılayamaz ve bizde zaten bu kadar veri kadar yüzlerce kez dosya.read(); diye yazmayacağımıza göre bunu kısaltmak gerekmektedir.

* **dosya.available();**

Bu fonksiyon dosyamızın sonuna gelip gelmediğimizi anlamak için kullanılmaktadır, dosyada okunabilecek karakter olduğu sürece 1 sonucunu geri döndürür, okunacak karakter kalmayınca 0 sonucunu geri döndürür. Bu sayede dosyanın sonuna geldiğimizi algılayamayan dosya.read(); okuma fonksiyonunun bu eksikliğini kapatabiliriz. Peki bu iki fonksiyonu satırlarca yazmadan nasıl kısaltacağız? Aşağıdaki kod penceresinde buna bir örnek görülmektedir.

int bilgi; // okunacak veriyi tutacak değişken tanımlandı

dosya = SD.open("LRT.txt",FILE\_READ); // LRT.txt adlı dosya okumak için açıldı

while (dosya.available()) // dosyada okunabilecek karakter olduğu sürece while çalışır

{

 bilgi = dosya.read(); // her while döngüsünde dosyadaki karakterler bitene

                       // kadar tek tek karakterler okunur ve bilgi adlı değişkene aktarılır

}

dosya.close(); // okunan dosya kapatılıyor.

Fakat yukarıdaki örneğin büyük bir eksiği var bu şekilde dosyadaki en son okunan karakteri elde etmiş oluruz, çünkü dosya.read(); fonksiyonu sadece bir ve her seferinde bir sonraki karakteri okumaktadır.

Arduino’nun da belli sınırları var. Mesela dilediğimiz uzunlukta karakterleri tek seferde okuyamayız. Bunu Arduino’nun RAM belleği sınırlıyor. Sadece boş bir SD kart kodu bile RAM ve Program belleğinin yaklaşık olarak yarısını (UNO için söyledim bunu) harcadığını düşünecek olursak biraz temkinli olmak gerekiyor.

Özetle Arduino bir dosya tarayıcısı veya okuyucusu değildir sınırımızı bilmemiz gerekli bu yüzden ihtiyacımız kadar veriyi okumalıyız. Verileri SD karttan okumanın en ideal yolu dizileri kullanmaktır. Ancak okunacak veriler Serial monitöre veya Bluetooth modül üzerinden telefona Serial.print ile gönderilecekse o zaman dizi değişkenlere ihtiyacımız olmayacaktır okunan her karakter anında Serial.print ile Serial monitöre veya telefonlara Bluetooth terminal programlarına aktarılabilir. Bunun avantajı Arduino üzerinde gereksiz yere RAM veya Program belleği alanı harcamamış oluşumuz hemde işlem yükünü hafifletmemizdir. Ancak dezavantaj olarak bu şekilde okunan verilerle işlem yapamıyor oluşumuzdur.

Aşağıda dizi kullanmadan okunan verileri olduğu gibi Serial monitöre aktaran bir program örneği bulunmaktadır.

dosya = SD.open("LRT.txt",FILE\_READ); // LRT.txt adlı dosya okumak için açıldı

while (dosya.available()) // dosyada okunabilecek karakter olduğu sürece while çalışır

{

 Serial.print((char)dosya.read()); // her while döngüsünde dosyadaki karakterler bitene

                       // kadar tek tek karakterler okunur ve Serial monitöre aktarılır

}

dosya.close(); // okunan dosya kapatılıyor.

Yukarıdaki örnekte dosya.read() fonksiyonu önünde (char) diye bir ibare var. Bunun anlamı şudur okunan veriyi karakter tipine dönüştür. Eğer ben bu ibareyi koymazsam o zaman Serial.print Serial monitöre ASCII tablosunda okunan her bir karakterin onluk karşılığını yan yana yazar buna Yeni Satır ve Taşıyıcı Dönüşü ifadeleri de dahildir! Bu ifadeler RS-232 protokolü içerisinde kullanılan komutlardır ve bir kaç tane daha vardır.  
“\n\r” şeklinde yazılan bu iki komutunda ASCII tablosunda onluk karşılığı bulunmaktadır. Ancak Karakter değildirler. ASCII tablosunda CR ve NL(LF) isimleri altında bakabilirsiniz.

Aşağıda ise okunan verilerin bir diziye aktarılarak Serial monitöre aktarılmasına ait örnek kod parçası görülmektedir. Bu kod gözünüzü korkutabilir ancak en kısa hali bu şekilde olmaktadır.

char okunan[100]; // karakter tipinde 100 karakterlik boş bir dizi tanımlanıyor

int i=0;

dosya = SD.open("LRT.txt",FILE\_READ); // okumak için dosya açılıyor

while (dosya.available()) // dosyada karakter olduğu sürece while döngüsü devam eder

{

okunan = dosya.read(); // i değişkeni başlangıçta 0 her döngüde bir sonraki karakter okunup diziye aktarılacak

i++; // bu sayede bir sonraki dizi adresine bilgi kaydedilmiş olacak

if(i==100) // 100 karakterlik dizi oluşturduğumuzdan bunu sınırlamalıyız

{

break; // 100 karakter okunduktan sonra while döngüsünden çıkılır

}

}

for(i=0;i<100;i++) // for döngüsü ile diziye kaydedilen her karakter sırayla Serial monitöre aktarılıyor

{

Serial.print(okunan);

}

dosya.close(); // okuma ve yazma işlemleri bitince dosya kapatılmalı.

Biz 100 adet karakter sınırında bir dizi tanımlamış olmamıza rağmen while içerisinde okunacak karakter miktarını if bloğu ile sınır koymasaydık o zaman Arduino içerisinde RAM bellek taşması gerçekleşip otomatik olarak Arduino kendini sıfırlayıp kendini baştan başlatması gerekecekti. Çünkü bizim dosyanın içerisinde 1000 belki 2000’den fazla karakter bulunuyor olabilir Arduino bunu anlamaz okuyabildiği kadar okur ancak sınır koyana kadar. Maalesef Serial.print ile okunan bir dizi tek bir kod satırı ile yazdırılmıyor. Her ne kadar dizimizin değişken tipi char yani karakter tipinde olsa da ne kadar içine okuyup aktardığımız veriler karakter verileri olsalar da Arduino bu işi yapamıyor. Nasıl ya Serial.print(“Merhaba LRT”); yazınca bal gibi de yazıyor diyebilirsiniz ancak arka planda olanlar farklı!

Örnek olarak;

Serial.print("Merhaba LRT");  // yazdığınızda arka planda olan şey aşağıdakine benzerdir ///////////////////////////////////////////////////////////////////////////////////////////////   
char text[]="Merhaba LRT";  // Merhaba LRT bir char tipinde dizi değişkene aktarılır   
for(int i=0; i<12; i++) // karakter sayısnın bir fazlası kadar for döngüsünde tek tek harfler yazdırılır   
{    
  Serial.print(text);   
}

Buraya kadar bir SD kartta dosya oluşturmayı üzerine veri yazmayı ve temel olarak veri kaydedip açılan dosyayı kapatmayı öğrendik. Ancak eğer kaydettiğimiz verileri SD karttan okuyup işlem yapmak istiyorsak o zaman bu okuduğumuz bilgileri sayısal değerlere dönüştürmemiz gerekmektedir. Mesela bir analog ölçüm yaptık ve bunu SD karta kaydettik. Bunu okursak elde edeceğimiz veri karakter dizisi olacaktır sayısal değil. Bundan dolayı bir dönüşüm gerçekleştirmemiz gerekmektedir. Bir sonraki başlığın konusu bunu kapsıyor olacak.

**SD KARTTAN OKUNAN SAYISAL VERİLERİN KARAKTERDEN SAYISAL İFADEYE DÖNÜŞÜM YÖNTEMLERİ**

Bunun için en basitinden iki yol var. Bunun için “stdlib.h” Standart C++ kütüphane fonksiyonları olan atoi() , atol() ve atof() fonksiyonlarını kullanmak bu yöntemlerden biridir. Bir diğer yöntem ise String adlı Class altında bir nesne oluşturup bu String nesnesi ile beraber toFloat() ve toInt() fonksiyonları ile karakter dizilerini sayısal hale getirmek bir diğer yöntemdir.

**YÖNTEM 1 – atoi() atol() ve atof() Fonksiyonları ile Rakamsal Karakter Dizilerini Sayısal Değere Dönüştürme**

* **atoi(parametre);**

Bu fonksiyon ile bir karakter dizisi içerisinde harf karakterlerini görmezden gelerek soldan sağa doğru tüm rakamsal karakterleri yan yana dizerek bunu int değişken tipinde  tam sayı haline çevirmektedir. parametre olarak karakter dizisi şeklinde bir char veri tipinde bir değişken olmalıdır.

**Örnek;**

char karakter[]="129abc4d0e"; // harf ve rakam karakterleri karışık bir karakter dizisi tanımlandı.

int sayi=atoi(karakter); // atoi ile bu dizi sayısal değere dönüştürülüyor.

Serial.println(sayi); // çıktı olarak ekranda 12940 görülecektir.

* **atol(parametre);**

Bu fonksiyon atoi() ile aynı işlemi yapmaktadır, tek fark atoi() dönüştürdüğü tam sayı değişken tipi int, bunda ise long tipindedir. int 16bit bir tam sayı değişken tipidir, long ise 32bit bir tam sayı değişken tipidir. Parametre olarak da atoi() fonksiyonunda olduğu gibi char değişken tipinde karakter dizisidir.

**Örnek;**

char karakter[]="139abc4d0e8fg"; // harf ve rakam karakterleri karışık bir karakter dizisi tanımlandı.

long sayi=atol(karakter); // atoi ile bu dizi sayısal değere dönüştürülüyor.

Serial.println(sayi); // çıktı olarak ekranda 139408 görülecektir.

* **atof(parametre);**

Bu fonksiyon atoi() ve atol() ile aynı işlemi yapmaktadır, tek fark atoi() ve atol() dönüştürdüğü tam sayı değişken tipi ondalıklı sayılar içindir yani float değişken tipindedir. float 32bit bir ondalıklı tam sayı değişken tipidir, long ise 32bit bir tam sayı değişken tipidir. Parametre olarak da atoi() ve atol() fonksiyonlarında olduğu gibi char değişken tipinde karakter dizisidir fakat bu sefer bu karakterlerden biri nokta olmalıdır aksi halde sayının ondalıklı kısmı sıfır olarak görülecektir yani sadece tam kısmını elde edebiliriz.

**Örnek;**

char karakter[]="13.0abc9d1e99f2g"; // harf ve rakam karakterleri karışık bir karakter dizisi tanımlandı.

long sayi=atol(karakter); // atoi ile bu dizi sayısal değere dönüştürülüyor.

Serial.println(sayi); // çıktı olarak ekranda 13.09 görülecektir.

Eğer ondalıklı kısmın basamak sayısını daha fazla görmek istiyorsanız Serial.println veya Serial.print içerisinde yazdırılacak ondalıklı tam sayının yanına virgül koyarak ondalıklı kısmın kaç adet basamağının yazdırılacağını belirtmemiz gerekmektedir.

**Örnek;**

char karakter[]="13.0abc9d1e99f2g"; // harf ve rakam karakterleri karışık bir karakter dizisi tanımlandı.

long sayi=atol(karakter); // atoi ile bu dizi sayısal değere dönüştürülüyor.

Serial.println(sayi,6); // çıktı olarak ekranda 13.091992 görülecektir.

Şimdi bunları SD Karttan okunan bir veriye uyarlayalım.  
   
Öncelikle SD kartımızda Arduino ile LRT.txt adında bir doysa oluşturalım..   
Bu dosyanın içine Arduino ile “123.456789” sayısını kaydedelim.  
Daha sonra bu veriyi okuyup okunan bu sayının iki katını alıp Serial monitöre yazdıralım.  
   
Kodu yazmadan önce bilmemiz gereken iki şey var;   
   
Birincisi kaydedeceğimiz bu ondalıklı sayı noktası dahil toplamda 10 karakter olduğu fakat sonun “\n\r” şeklinde iki adet CR ve NL(LF) adlı ASCII komutları eklendiğinden biz bu ondalıklı sayıyı kaydettiğimizde toplamda 12 karakterlik yer kaplayacaktır.  
   
İkincisi kaydedilen ve okunacak olan bu sayı float yani ondalıklı tam sayı olduğundan dönüşüm için atof() fonksiyonu kullanılacaktır.  
   
atoi(), atof() ve atol() fonksiyonları “stdlib.h” Standart C++ kütüphane fonksiyonları olduğundan bulundukları kütüphaneyi kodumuzda dahil etmek zorunda değiliz otomatik olarak yazmasak bile dahil edilir.  
   
Kod aşağıdaki gibidir.

#include <SPI.h> // SPI kütüphanesi eklendi

#include <SD.h> // SD kart kütüphanesi eklendi

File dosya; // dosya adında bir File nesnesi oluşturuldu

void setup()

{

Serial.begin(115200); // SD karttan okunan verileri Serial monitöre daha hızlı aktarmak için 115200baud hızı seçildi

// aynı değişikliği Serial monitörünüzde önceden yapmayı unutmayın

Serial.println("SD kart hazirlaniyor...");

if(!SD.begin(4)) // SD kart CS bacağı 4 nolu dijital pine bağlı, SD kart takılı değilse

{

Serial.println("SD kart takili degil!!!");

Serial.println("Lutfen SD kartinizi takiniz!!!");

while(!SD.begin(4)); // kart takılana kadar beklenir

Serial.println("SD kart kullanima hazir!!!"); // kart takılınca program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

else

{

Serial.println("SD kart kullanima hazir!!!"); // kart takılı ise program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

}

void loop()

{

Serial.println("LRT SD KART TEST");

char okunan[100]; // 100 karakterlik char tipinde boş bir dizi tanımlandı

int i=0; // okunacak karakter sayısı için sınır değişken başlangıçta sıfır

dosya = SD.open("LRT.txt",FILE\_WRITE); // LRT.txt adlı bir dosya üzerine veri yazılmak üzere oluşturulup açılıyor.

dosya.println(123.456789,6); // açılan dosyaya 123.456789 sayısı yazılıyor dondaki 6 ondalıklı

// kısmın kaç basamağının yazılacağını belirtiyor

dosya.close(); // yazma işlemi bitti dosya kapatılıyor

Serial.println("123.456789 sayisi SD Karta Yazildi...");

dosya = SD.open("LRT.txt",FILE\_READ); // aynı dosya bu sefer okunmak için açılıyor

while (dosya.available()) // dosyada karakter olduğu sürece while döngüsü devam ediyor

{

okunan[i] = dosya.read(); // ilk karakterden başlayıp her döngüde bir sonraki karakteri

// boş dizi değişkenine aktarıyoruz

i++;

if(i==12) // 12 karakter sınırı koyduk bundan fazla karakter okunmayacak dosyadan

{

break; // okunmaya kalkışılırsa while döngüsü sonlandırılır programa devam edilir

}

}

dosya.close(); // dosya kapatılıyor

float okunan\_rakam = atof (okunan); // okunan karakter dizisi float tipinde tam sayıya dönüştürülüyor

Serial.println(okunan\_rakam,6); // ekrana dosyadan okunan ve float tipinde tam sayıya çevrilmiş hali

// Serial monitöre yazdırılıyor, ondalıklı kısım 6 basamaklı olacak şekilde.

// çıktı 123.456771 oluyor gerçekte kaydedilen değer ise 123.456787 oluyor

while(1); // program sonu loop tekrarını istemiyorum şimdilik

}

Bu kod ile ilgili konuşalım. Üstelik sondan başlayalım. Öncelikle kaydetmeye çalıştığımız rakam “123.456789”. Dosyaya kaydedilen “123.456787”.  
Dosyadan okunup karakter dizisinden float tam sayı tipine dönüştürüp Serial monitörde görülen ise “123.456771”.   
   
Bunun sebebi Arduino’nun ondalıklı sayıları hesaplayabilen bir aritmetik işlem yöntemine sahip olmayışıdır. Arduino sadece 8bit veya 1byte veriler ile işlem yapabilmektedir. Bundan dolayı ondalıklı sayılarla işlem yapıldıkça hatalar oluşmaktadır. Önce SD karta yazarken oluşan bir hata var “123.456789” sayısı “123.456787” şeklinde değişiyor. Daha sonra SD karttan bu yazılan bilgi tekrar okunurken de bir hata oluşuyor ve Serial monitörde “123.456771” olarak görülüyor. Hata 10000’de 1 olsa bile hata hatadır.

Bu yüzden SD karttan veri okuyup işlem yapılmasını tavsiye etmiyorum. Siz verilerinizi SD  karta kayıt edin ancak işlenmesini Bilgisayar ortamında yapınız. Sonra günlerce hesaplamalarda nerede hata var diye kendinizi harcarken bulursanız şaşırmayın.  
   
Buraya kadar da herşey güzel artık bir SD kattan kayıt edilen bir bilgiyi okuyup nasıl tekrar sayısal değere dönüştürdüğümüzü de öğrendik. Fakat hala bir eksik var. Biz SD karta ve bir dosyaya sadece bir adet veri kaydetmiyoruz. Yüzlerce veriyi kaydettik peki hangisine nasıl ulaşıyoruz? Bunun için bir fonksiyon var ancak kullanımı çok rahat değil ancak iş görüyor çünkü bu tek yöntem.

* **dosya.seek(parametre);**

Bu fonksiyon aslında bir imleç görevi görüyor. Parametresine hangi rakamı yazarsanız örneğim 10 yazalım, o zaman imleç yani dosyaya verinin yazılacağı veya okunacağı nokta 11’ci karakterden itibaren olacaktır. Çünkü aynı dizi mantığı gibi, nasıl 0 indisiyle bir diziden 1’ci eleman seçiliyorsa burada da parametreye 0 yazınca dosyanın 1’ci karakterinden itibaren imleç konumlanır ve imleç neredeyse oradan itibaren veri okumaya veya yazılmaya devam edilir.  
   
Örnek olarak LRT\_SD.txt adında bir dosya oluşturup içine;

123456789​

234567891​

345678912​

456789123​

567891234​

yazıp kaydedelim ardında SD kartımıza bilgisayardan içine atalım sonra SD kart modülümüze SD kartımızı takalım ve aşağıdaki örnek kodu upload edelim.  
   
Yapacağımız işlem basit, sadece 3 ile başlayan satırı, yani “345678912” , SD karttan okuyup atol() ile long tipinde bir tam sayıya çevirip Serial monitöre yazdırmak.

#include <SPI.h> // SPI kütüphanesi eklendi

#include <SD.h> // SD kart kütüphanesi eklendi

File dosya; // dosya adında bir File nesnesi oluşturuldu

void setup()

{

Serial.begin(115200); // SD karttan okunan verileri Serial monitöre daha hızlı aktarmak için 115200baud hızı seçildi

// aynı değişikliği Serial monitörünüzde önceden yapmayı unutmayın

Serial.println("SD kart hazirlaniyor...");

if(!SD.begin(4)) // SD kart CS bacağı 4 nolu dijital pine bağlı, SD kart takılı değilse

{

Serial.println("SD kart takili degil!!!");

Serial.println("Lutfen SD kartinizi takiniz!!!");

while(!SD.begin(4)); // kart takılana kadar beklenir

Serial.println("SD kart kullanima hazir!!!"); // kart takılınca program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

else

{

Serial.println("SD kart kullanima hazir!!!"); // kart takılı ise program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

}

void loop()

{

Serial.println("LRT SD KART TEST");

char okunan[100]; // 100 karakterlik char tipinde boş bir dizi tanımlandı

int i=0; // okunacak karakter sayısı için sınır değişken başlangıçta sıfır

dosya = SD.open("LRT\_SD.txt",FILE\_READ); // LRT.txt adlı bir dosya üzerine veri yazılmak üzere oluşturulup açılıyor.

dosya.seek(22); // 3 ile başlayan satırın il karakterine ait imleç pozisyonu okuma işlemi buradan başlayacak.

while (dosya.available())

{

okunan[i] = dosya.read(); // karakterler tek tek dizi değişkene sırayla aktarılıyor

i++;

if(i==9) // sınır 9 karakter olarak belirlendi çünkü her satırda 9 karakterli rakam verisi var.

{

break;

}

}

dosya.close(); // dosya kapatılıyor

long okunan\_rakam = atol (okunan); // okunan karakterler long tipinde tam sayıya dönüştürülüyor

Serial.println(okunan\_rakam); // Serial monitöre okunan ve dönüştürülmüş olan sayı yazdırılıyor. Çıktı "345678912"

while(1); // program sonu

}

Burada dikkat edilecek şey yine ASCII komutlarının unutulmamasıdır, CR ve NL(LF) komutları. Bunlar görünür karakterler olmamasına rağmen vardırlar. Bundan dolayı 1.satırından 2.satıra geçerken ve 2.satırdan 3.satırın il karakterinin pozisyonunu sayarken bu ASCII komutlarını da saymamız gerekiyor. Bu duruma her bir sonraki satıra geçerken dikkat edilmesi gerekmektedir aksi halde bilginin bir kısmını es geçebilirsiniz.

Eğer dosyanın en başından 3. satırın başına 0’dan başlayıp sayarsanız ve her satır atlarken CR ve NL(LF) ASCII komutlarını da ekleyerek devam ederseniz 22 saymış olacaksınız bunu da dosya.seek() fonksiyonu parametresi yerine yazmış oluyoruz.Tabi bu sayma işlemi yukarıdaki örnek olarak kaydettiğimiz 5 satırlık 9 basamaklı rakamlar için geçerlidir. Farklı dosya içeriklerine göre hangi satırda hangi yerden veriyi okumak istiyorsanız ne yazık ki o noktanın imleç pozisyonu sayısını bilmeniz gerekmektedir.  
   
Dikkat ederseniz örnekteki dosyada her satırdaki rakamlar 9 basamaklı tam sayılardır ve bunlar int türünde bir değişkene sığmayacağı için long tipinde tam sayı değişkene dönüştürmem gerektiği için atol() fonksiyonu kullanıldı.   
   
İmleç pozisyonunu yani konumunu veren bir fonksiyon da var bu sayede imlecin nerede hangi pozisyonda kaldığını öğrenebilirsiniz. Bu fonksiyon aşağıdaki gibidir.

* **dosya.position();**

  Geri dönüş tipi işaretsiz unsigned long tipinde bir tam sayıdır. Ancak işaretli long tipinde bir tam sayının pozitif değerlikli sınırı bile yeterince büyük olduğunda long tipinde bir değişkene eşitleyerek de kullanabilirsiniz bu fonksiyonu.

**Örnek;**

**long pozisyon = dosya.position();**

**YÖNTEM 2 – String Class Nesnesi ve Fonksiyonları ile Rakamsal Karakter Dizilerini Sayısal Değere Dönüştürme**

Yöntem 1 ile yapılan işlem arasında mantıksal olarak bir fark görünmemektedir. Hatta Kodlarda bir kısalma da olmamaktadır. Yöntem 1 ile arasındaki en büyük fark String Class ile String nesnesi oluşturuyor olmamızdır. Yöntem 1 de dizi oluşturuyorduk ve buna bir sınır koyuyorduk. Bunda oluşturulan String nesnesine karakter sınırı koymuyoruz. Sınırlama tamamen bu nesneye aktarılan karakter miktarı ile belirlenir. Eğer 9 karakter aktarılacak ise o zaman String nesnemiz 9 elemanlı bir dizi olarak var olacaktır. Bir diğer farklılık ise dosya.read(); fonksiyonu başına (char) ifadesini koymamız gerektiğidir. Bunun sebebi toFloat veya toInt dönüşüm fonksiyonlarının String nesnesi içerisinde karakter tipinde değişkenler arıyor olmasıdır.

* **String okunan=””;**

String Class içerisinde string türünde okunan adlı boş bir string değişken tanımlanıyor.

* **okunan.endsWith(parametre);**

Bu fonksiyon parametresine yazılan ASCII karakterini string değişken içerisinde bulursa sonucunu 1 bulamazsa 0 olarak geri dönüş değeri olarak vermektedir. Bu sayede okunan bir satırın içerisinde anahtar bir karaktere göre SD karttan okumanın kesilmesini sağlayabiliriz.

* **okunan.toInt();**

Bu fonksiyon ile okunan adlı string ifadeye aktarılan rakam karakterlerini atol() fonksiyonu gibi long tipinde tam sayıya dönüştürmektedir. toInt yazmasına rağmen long türünde bir değişkene dönüştürmektedir.

* **okunan.toFloat();**

Bu fonksiyon ile okunan adlı string ifadeye aktarılan ondalıklı rakam karakterlerini atof() fonksiyonu gibi float tipinde ondalıklı tam sayıya dönüştürmektedir.   Aslında tüm farklılık arka planda olsa da uygulanış olarak bir iki farklılık olmaktadır. Bunları Yöntem 1 sonundaki kodun Yöntem 2 ile uygulanışı olarak değiştirecek olursak örnek kod aşağıdaki gibi olacaktır. Aynı göründüğüne aldanmayın farklılar ve farklılıkları kendiniz bulmaya çalışın.

#include <SPI.h> // SPI kütüphanesi eklendi

#include <SD.h> // SD kart kütüphanesi eklendi

File dosya; // dosya adında bir File nesnesi oluşturuldu

String okunan=""; // okunan adında boş bir String nesnesi tanımladık.

void setup()

{

Serial.begin(115200); // SD karttan okunan verileri Serial monitöre daha hızlı aktarmak için 115200baud hızı seçildi

// aynı değişikliği Serial monitörünüzde önceden yapmayı unutmayın

Serial.println("SD kart hazirlaniyor...");

if(!SD.begin(4)) // SD kart CS bacağı 4 nolu dijital pine bağlı, SD kart takılı değilse

{

Serial.println("SD kart takili degil!!!");

Serial.println("Lutfen SD kartinizi takiniz!!!");

while(!SD.begin(4)); // kart takılana kadar beklenir

Serial.println("SD kart kullanima hazir!!!"); // kart takılınca program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

else

{

Serial.println("SD kart kullanima hazir!!!"); // kart takılı ise program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

}

void loop()

{

Serial.println("LRT SD KART TEST");

int i=0; // okunacak karakter sayısı için sınır değişken başlangıçta sıfır

dosya = SD.open("LRT\_SD.txt",FILE\_READ); // LRT.txt adlı bir dosya üzerine veri yazılmak üzere oluşturulup açılıyor.

dosya.seek(22); // 3 ile başlayan satırın ilk karakterine ait imleç pozisyonu okuma işlemi buradan başlayacak.

while (dosya.available())

{

okunan += (char)dosya.read(); // karakterler tek tek string nesnesine sırayla aktarılıyor

i++;

if(i==9) // sınır 9 karakter olarak belirlendi çünkü her satırda 9 karakterli rakam verisi var.

{

break;

}

}

dosya.close(); // dosya kapatılıyor

long okunan\_rakam = okunan.toInt(); // okunan karakterler long tipinde tam sayıya dönüştürülüyor

Serial.println(okunan\_rakam); // Serial monitöre okunan ve dönüştürülmüş olan sayı yazdırılıyor. Çıktı "345678912"

while(1); // program sonu

}

Yöntem 2 ile verileri SD karttan okumanın bir avantajı var.

Örneğin onlarca satırlık bir analog dijital çevirici sonucu listesi kaydettiniz ve bunlardan dilediğiniz satırdaki ölçüme ulaşmanız gerekecek. String nesnesi içerisine NL(LF) ASCII komutu yazılana kadar içerisine karakterler aktarılabilmektedir. Böylece yanlışlıkla birden fazla satırdaki verileri okuma durumuna düşmezsiniz. Bunu doğrulamak için ise o an okunan karakterlerin bulunduğu satırda NL(LF) ASCII komutunun gelip gelmediğini kontrol etmek gerekir. Bu kontrol işlemini endsWith(); fonksiyonu yardımıyla yapabiliriz.  
   
Aşağıdaki kod buna bir örnektir.

#include <SPI.h> // SPI kütüphanesi eklendi

#include <SD.h> // SD kart kütüphanesi eklendi

File dosya; // dosya adında bir File nesnesi oluşturuldu

String okunan=""; // okunan adında boş bir String nesnesi tanımladık.

void setup()

{

Serial.begin(115200); // SD karttan okunan verileri Serial monitöre daha hızlı aktarmak için 115200baud hızı seçildi

// aynı değişikliği Serial monitörünüzde önceden yapmayı unutmayın

Serial.println("SD kart hazirlaniyor...");

if(!SD.begin(4)) // SD kart CS bacağı 4 nolu dijital pine bağlı, SD kart takılı değilse

{

Serial.println("SD kart takili degil!!!");

Serial.println("Lutfen SD kartinizi takiniz!!!");

while(!SD.begin(4)); // kart takılana kadar beklenir

Serial.println("SD kart kullanima hazir!!!"); // kart takılınca program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

else

{

Serial.println("SD kart kullanima hazir!!!"); // kart takılı ise program devam eder

delay(1000); // 1 saniye isteğe bağlı bekleme

}

}

void loop()

{

Serial.println("LRT SD KART TEST");

dosya = SD.open("LRT\_SD.txt",FILE\_READ); // LRT.txt adlı bir dosya üzerine veri yazılmak üzere oluşturulup açılıyor.

dosya.seek(0); // 1 ile başlayan satırın ilk karakterine ait imleç pozisyonu okuma işlemi buradan başlayacak.

while (dosya.available())

{

okunan += (char)dosya.read(); // karakterler tek tek dizi değişkene sırayla aktarılıyor

if(okunan.endsWith("\n")) // sınır 20 karakter olarak belirlendi çünkü her satırda 9 karakterli rakam verisi var.

{

break;

}

}

dosya.close(); // dosya kapatılıyor

long okunan\_rakam = okunan.toInt(); // okunan karakterler long tipinde tam sayıya dönüştürülüyor

Serial.println(okunan\_rakam); // Serial monitöre okunan ve dönüştürülmüş olan sayı yazdırılıyor. Çıktı "123456789"

while(1); // program sonu

}

Böylece artık SD kart içerisine kaydedilen veya kayıt edilmiş verilerin nasıl okunup sayısal değerlere karakter tipinden dönüştürüldüğünü öğrenmiş olduk. Çok uzun bir başlık oldu ve daha yazılması gereken çok şey var ancak bunlar yeterli olacaktır.

**SD KART İÇERİSİNDE DOSYA SİLME KLASÖR AÇMA VE KLASÖR SİLME**

SD kartınızda klasörler oluşturabilir dilerseniz bu oluşturduğunuz klasörler içerisinde dosya oluşturabilirsiniz. Örneğin A sensörünün verilerini SensorA diye bir klasörde VeriA adında bir dosya içerisinde kaydedebilirsiniz.

* **SD.mkdir(klasöryolu);**

Bu fonksiyon ile SD kart içerisine klasörler ve hatta iç içe bir kaç tane klasör oluşturabilmemizi sağlamaktadır. Kullanırken herhangi bir dosyanın açık olmamasına dikkat edilmelidir, derleme esnasında sorun yaşamayabilirsiniz ancak program çalışırken sorunlar meydana gelecektir. Fonksiyonu akılda tutmak için ” mkdir ” isminin “make directory” ifadesinin yani “klasör oluştur” kısaltılması olduğunu unutmayın.  
   
Klasör yolu ise “a/b/c” şeklinde ifade edilir mesela bu şekilde iç içe a b ve c adında klasörler oluşturmuş olursunuz. Eğer sadece “a” yazılsaydı o zaman sadece a adında bir klasör oluşturulurdu.

**Örnek;**

**SD.mkdir("a/b");**// iç içe a sonra b klasörü oluşturulur.​

* **SD.rmdir(klasöryolu);**

Bu fonksiyon ile SD kart içerisinde daha önceden oluşturulmuş olan klasörleri silebiliyoruz. Ancak klasör oluştururken birden fazla iç içe klasör oluşturur gibi iç içe olan klasörleri silemiyoruz. Hatta eğer silmek istediğimiz klasörün içinde bir dosya var ise yine bu klasör silinemez. Kullanırken herhangi bir dosyanın açık olmamasına dikkat edilmelidir, derleme esnasında sorun yaşamayabilirsiniz ancak program çalışırken sorunlar meydana gelecektir. Fonksiyonu akılda tutmak için ” rmdir ” isminin “remove directory” ifadesinin yani “klasör sil” kısaltılması olduğunu unutmayın.  
  
Klasör yolu olarak yazılan , örneğin “a/b/c” yazılırsa , en sondaki c klasörünün silineceği anlamına gelecektir. Eğer iç içe a, b ve c adında 3 klasör var ise ve klasör yolu olarak “a” yazarsak bu üç klasör iç içe olmasına rağmen hiç biri silinmeyecektir çünkü silinecek klasörün içerisinde ne bir klasör ne de bir dosyanın olmaması gerekmektedir.

**Örnek;**

**SD.rmdir("a/b");**// diyelim A içinde B, B içinde de C klasörümüz var ve C klasörünün içi boş olsun ve bu fonksiyon ile B klasörünü silmeye kalkışırsak B içinde C klasörü olduğundan silme işlemi gerçekleşmeyecektir. Bu yüzden önce C klasörü silinmelidir daha sonra B klasörü silinmelidir.​

**SD.rmdir("a/b/c");** // C klasörü siliniyor​

**SD.rmdir("a/b");**// B klasörü siliniyor​Bu şekilde önce C klasörü silinmiş olacaktır daha sonra ise B klasörü silinecektir.​

* **SD.remove(dosyayolu);**

Bu fonksiyon ile silmek istediğimiz bir dosyayı silebiliriz. Eğer dosya bir klasör veya iç içe herhangi bir klasörde ise o zaman dosya yolu ile birlikte dosya adı ve uzantısını birlikte yazarak silme işlemini gerçekleştirebiliriz.

**Örnek;**

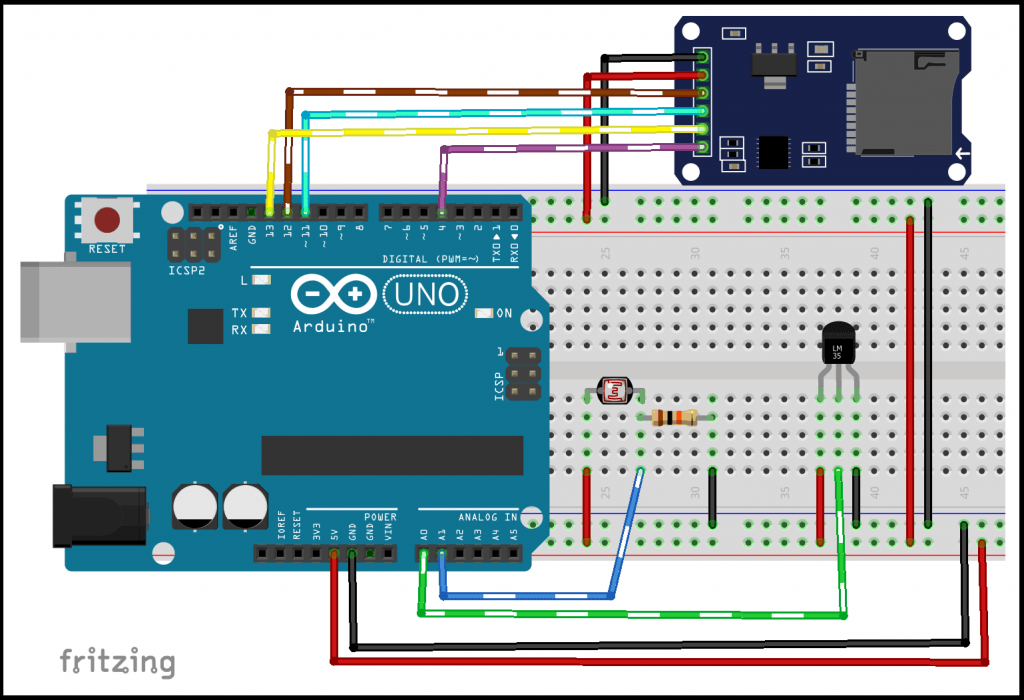
**SD.remove("a/b/c/LRT\_SD.TXT");**​// Diyelim A içinde B, B içinde ise C klasörü olsun ve bu C klasöründe de LRT\_SD.TXT adında bir dosya olsun. Ben Bu dosyayı silmek için Yukarıdaki örnek kod satırı ile "a/b/c/LRT\_SD.TXT" şeklinde dosya yolunu belirerek dosya adı ve uzantısı ile birlikte silme işlemini gerçekleştirmiş olurum.​

**SD.remove("LRT\_SD.TXT");**​// Eğer dosya herhangi bir klasör içerisinde değilse o zaman dosya yolu olarak sadece dosya adı ve uzantısını yazmak yeterli olacaktır.​

  Artık bir SD kartın nasıl kullanılması gerektiğini kütüphane fonksiyonlarına göre öğrendik gelin şimdi bir de bunu örnek bir mini proje olarak uygulamasını yapalım. Unutmayın SD kart ile ilgili sizlere sadece gerekli olabilecek fonksiyonlardan bahsettim. Bu konuyu bu kadar irdeledim çünkü Arduino IDE içindeki SD kart örnek kodları çok fazla anlaşılır gelmedi bana test edip sonuçlarını göre göre neyin ne işe yaradığını kavradım. Bende edindiğim bilgileri günü gününe kaydedip sonrasında buraya konu olarak aktardım.

**ARDUINO İLE SD KART DATALOGGER UYGULAMASI**

Projemizde bir adet LDR ve bir adet LM35 sıcaklık sensörü kullanılacak. Bu proje ile LDR den gelen analog bilgiyi saniyede bir toplamda 6 kere, LDR adında bir klasör içerisinde LDR.txt adında bir dosyaya kaydedeceğiz. Aynı zamanda yine 1 saniye aralıkla toplamda 6 kere LM35 ile ölçülen sıcaklığı LM35 adında bir klasör içerisinde LM35.txt adında bir dosyaya kaydedeceğiz. Daha sonra LM35 ve LDR ile elde edilip kaydedilen bu bilgileri SD karttan okuyup 2 katını Serial monitöre yazdıracağız. Fritzing şeması aşağıdaki gibidir.



**MALZEME LİSTESİ**

* SD KART Modülü ( microSD veya SD girişli olması çok fark etmemekte )
* 1 Adet microSD veya SD kart ( microSD kart alanların SD adaptörlü olarak almalarını tavsiye ederim )
* 1 Adet LM35 sıcaklık sensörü
* 1 Adet 10K Ohm LDR
* 1 Adet 10K Ohm direnç
* 1 Adet Arduino UNO, NANO veya MICRO ( 3’nün de bacak numaraları aynıdır )
* 1 Adet Breadboard
* Yeterince Jumper veya tek damarlı zil teli
* Sabır

**ARDUINO KODU**

|  |  |
| --- | --- |
|  | /\* |
|  | \* Lezzetli Robot Tarifleri |
|  | \* https://lezzetlirobottarifleri.com/arduino-ile-sd-kart-kullanimi/ |
|  | \*/ |
|  |  |
|  | #include <SPI.h> // SPI kütüphanesi tanımlanıyor |
|  | #include <SD.h> // SD kart kütüphanesi tanımlanıyor |
|  |  |
|  | File dosya; // dosya adında bir File nesnesi oluşturuluyor |
|  |  |
|  | String okunan\_lm35="", okunan\_ldr=""; // String türünde iki adet değişken tanımlanıyor |
|  |  |
|  | int LM35 = A0; // analog pin 0 LM35 bağlanıyor |
|  | int LDR = A1; // analog pin 1 LDR bağlanıyor |
|  |  |
|  | unsigned long eski\_zaman; // delay olmadan bekleme oluşturmak için gerekli değişkenler |
|  | const long bekleme\_suresi=1000; // 1 saniye yani 1000 milisaniyelik bekleme süresi tanımlanıyor |
|  |  |
|  | int kayit = 0; // okunan kayıtların sayısını tutuan değişken |
|  | int max\_kayit = 6; // okunacak maksimum kayit mikatrı sınırı için değişken |
|  |  |
|  | // LM35 sıcaklık ölçümü fonksiyonu |
|  | float LM35\_oku() |
|  | { |
|  | int bilgi = analogRead(LM35); |
|  | float sicaklik = bilgi \* 0.48828125; |
|  | return sicaklik; |
|  | } |
|  |  |
|  | // LDR analog ölçüm fonksiyonu |
|  | long LDR\_oku() |
|  | { |
|  | int bilgi = analogRead(LDR); |
|  | float isik = bilgi; |
|  | return isik; |
|  | } |
|  |  |
|  | void setup() |
|  | { |
|  | Serial.begin(115200); // 115200baud hızında serial iletişim başlatılıyor |
|  |  |
|  | Serial.println("SD KART DATALOGGER UYGULAMASI"); |
|  |  |
|  | // SD kartın takılıp takılmadığını kontrol eden program bloğu |
|  | // takılı değilse SD kart takılana kadar bekler |
|  | // takılınca program kaldığı yerden devam eder |
|  | // SD.begin(4) yazıldığı için 4 nolu dijital pin SD kart modülünün CS bacağına bağlanmalıdır |
|  | Serial.println("SD kart hazirlaniyor..."); |
|  | if(!SD.begin(4)) |
|  | { |
|  | Serial.println("SD kart takili degil!!!"); |
|  | Serial.println("Lutfen SD kartinizi takiniz!!!"); |
|  | while(!SD.begin(4)); |
|  | Serial.println("SD kart kullanima hazir!!!"); |
|  | delay(1000); |
|  | } |
|  | else |
|  | { |
|  | Serial.println("SD kart kullanima hazir!!!"); |
|  | delay(1000); |
|  | } |
|  |  |
|  | SD.mkdir("LM35"); // LM35 ölçüm sonuçlarının kaydedileceği klasör oluşturuluyor |
|  | SD.mkdir("LDR"); // LDR ölçüm sonuçlarının kaydedileceği klasör oluşturuluyor |
|  | } |
|  |  |
|  | void loop() |
|  | { |
|  | // kod bloğu bekleme işlemini kontrol etmektedir. |
|  | // belirlenen bekleme süresi dolduğunda Ölçümler her iki sensör için |
|  | // bir kez alınır ve SD kartta ilgili klasörde kayıt edilir |
|  | // 6 ölçüm alındıktan sonra görevi biter |
|  | unsigned long simdiki\_zaman = millis(); |
|  | if(simdiki\_zaman - eski\_zaman >= bekleme\_suresi) |
|  | { |
|  | eski\_zaman = simdiki\_zaman ; |
|  |  |
|  | dosya = SD.open("LM35/LM35.txt",FILE\_WRITE); |
|  | if(dosya) |
|  | { |
|  | dosya.println(LM35\_oku()); |
|  | } |
|  | dosya.close(); |
|  |  |
|  | dosya = SD.open("LDR/LDR.txt",FILE\_WRITE); |
|  | if(dosya) |
|  | { |
|  | dosya.println(LDR\_oku()); |
|  | } |
|  | dosya.close(); |
|  |  |
|  | kayit++; |
|  | Serial.print("."); |
|  |  |
|  | } |
|  |  |
|  | // 6 adet ölçüm alınıp SD karta kaydedildikten sonra bu ölçümler SD karttan okunur |
|  | if(kayit >= max\_kayit) |
|  | { |
|  | Serial.println("."); |
|  | // SD karttan okunan verilerin string ifadeden sayısal verilere dönüştürüldükten |
|  | // sonra kaydedilecek yerel dizi değişkenleri tanımlanıyor |
|  | int j=0; |
|  | float okunan\_deger\_lm35[10]; |
|  | long okunan\_deger\_ldr[10]; |
|  |  |
|  | dosya = SD.open("LM35/LM35.txt",FILE\_READ); // LM35 ölçümlerinin okunacağı dosya kendi klasöründe açılıyor |
|  | // 6 adet kayıt edilmiş ölçüm SD karttan alınana kadar while döngüsü devam eder |
|  | while (dosya.available()) |
|  | { |
|  | okunan\_lm35 += (char)dosya.read(); // her döngüde bir adet ölçümün tek tek her bir karakteri string değişkene aktarılır |
|  |  |
|  | // her bir ölçüm sonucu ayrı ayrı satırlardadır. satır sonu |
|  | //kontrolu için NL komutu var mı kontrol edilir. NL ASCII komutu yeni satıra geçmeden ilgili satırdaki son karakterdir. |
|  | if(okunan\_lm35.endsWith("\n")) |
|  | { |
|  | okunan\_deger\_lm35[j] = 2\*okunan\_lm35.toFloat(); // okunan ölçüm değer ondalıklı tam sayı değere dönüştürülür ve |
|  | // 2 katı olarak bir diziye kayıt edilir |
|  | okunan\_lm35=""; // bir sonraki ölçümün okuması için string değişken sıfırlanır. |
|  | j++; // sonraki dizi elemanına kayıt edilmek için indis bir arttırılır |
|  | } |
|  | if(j==6) // 6 adet ölçüm sayısal veriye dönüştürülüp iki katı alındıktan sonra while döngüsü sona erer. |
|  | { |
|  | j=0; |
|  | break; |
|  | } |
|  | } |
|  | dosya.close(); // okuma işlemi bitti LM35 dosyası kapatılır |
|  |  |
|  | // LM35 okuması ile aynıdır tek fark dönüştürülen sayı tam sayıdır. LM35 de ondalıklı tam sayıya dönüştürüldü. |
|  | dosya = SD.open("LDR/LDR.txt",FILE\_READ); |
|  | while (dosya.available()) |
|  | { |
|  | okunan\_ldr += (char)dosya.read(); |
|  | if(okunan\_ldr.endsWith("\n")) |
|  | { |
|  | okunan\_deger\_ldr[j] = 2\*okunan\_ldr.toInt(); |
|  | okunan\_ldr=""; |
|  | j++; |
|  | } |
|  | if(j==6) |
|  | { |
|  | j=0; |
|  | break; |
|  | } |
|  | } |
|  | dosya.close(); |
|  |  |
|  | // okunan ve iki katı olarak elde edilen 6 adet LM35 ve 6 adet LDR değerleri ekrana yazdırılır |
|  | for(int i=0;i<6;i++) |
|  | { |
|  | Serial.print("LM35 Deger "); |
|  | Serial.print(i+1); |
|  | Serial.print(": "); |
|  | Serial.print(okunan\_deger\_lm35[i]); |
|  | Serial.print(" // LDR Deger "); |
|  | Serial.print(i+1); |
|  | Serial.print(": "); |
|  | Serial.println(okunan\_deger\_ldr[i]); |
|  | } |
|  |  |
|  | while(1); // program burada durur devam etmez. |
|  | } |
|  | } |