LEER Y ESCRIBIR EN UNA TARJETA SD O MICRO SD CON ARDUINO

(VERSION: 16-7-19)

FUENTE: https://www.luisllamas.es/tarjeta-micro-sd-arduino/

¿QUÉ ES UN LECTOR SD O MICRO SD?

Un lector SD es un dispositivo que permite emplear como almacenamiento una tarjeta SD, que podemos incorporar en nuestros proyectos de electrónica y Arduino.

Las tarjetas SD y micro SD se han convertido en un estándar, desplazando a otros medios de almacenamiento de datos debido a su gran capacidad y pequeño tamaño. Por este motivo han sido integradas en una gran cantidad de dispositivos, siendo en la actualidad componentes frecuentes en ordenadores, tablets y smartphones, entre otros.

Dentro del mundo de Arduino, es posible encontrar lectores de bajo coste tanto para tarjetas SD como micro SD. Los primeros en aparecer fueron los lectores SD y posteriormente los micro SD. Por tanto, en general, los módulos con micro SD son modelos más modernos que los de SD

En ambos tipos de lectores, la lectura puede realizarse a través de bus SPI. Aunque pueden disponer de otros interfaces, como bus I2C o UART, normalmente es preferible emplear SPI por su alta tasa de transferencia.

Respecto a las tarjetas empleadas, podemos emplear tarjetas SD o SDSC (Standard Capacity) o SDHC (High Capacity), pero no SDXC (Extended Capacity). Deberá estar formateada en sistema de archivos FAT16 o FAT32.

La tensión de alimentación es de 3.3V, pero en la mayoría de los módulos se incorpora la electrónica necesaria para conectarlo de forma sencilla a Arduino, lo que frecuentemente incluye un regulador de voltaje que permite alimentar directamente a 5V.

Emplear una tarjeta SD o micro SD en con Arduino tiene la ventaja de proporcionar una memoria casi ilimitada para nuestros proyectos. Además es no volátil (es decir, resiste cuando se elimina la alimentación), y puede ser extraída y conectada a un ordenador con facilidad.

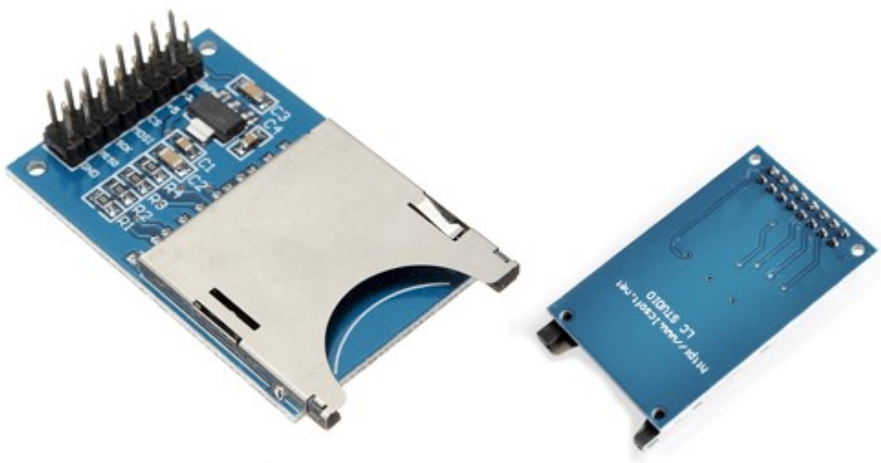
La gran desventaja es que supone una importante carga de trabajo para Arduino. Sólo el programa ocupará el 40% de la memoria Flash, y casi el 50% de la memoria dinámica. El uso del procesador también es exigente.

En general, manejar una tarjeta micro SD o SD puede considerarse en al límite de la capacidad de Arduino como procesador. Aunque es posible emplearlas en proyectos sencillos, en proyectos reales deberíamos plantearnos emplear otra opción como un Raspberry Pi.

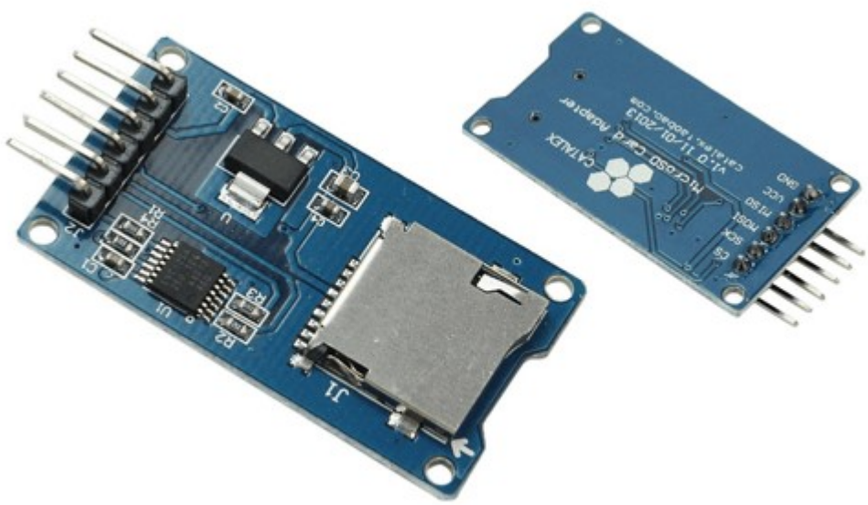
En los proyectos caseros se emplean tarjetas SD y micro SD, principalmente, en proyectos de tipo datalogger, es decir, para mantener el registro de las mediciones de un sensor. También pueden ser empleadas, por ejemplo, para llevar movimientos o rutas de robots precalculados, y cargarlos bajo dejo demanda.

PRECIO

Los lectores de tarjetas son dispositivos muy baratos. Podemos encontrar lectores de tarjetas micro SD por unos 0.35€, y lectores de tarjetas SD por unos 0.65€, buscando en vendedores internacionales de eBay o AliExpress.



Dado que los lectores de tarjetas micro SD son más baratos, modernos, y compactos, normalmente preferiremos los lectores micro SD a los SD. El único motivo real para comprar un lector SD frente a un micro SD es aprovechar alguna tarjeta que tengamos disponible.

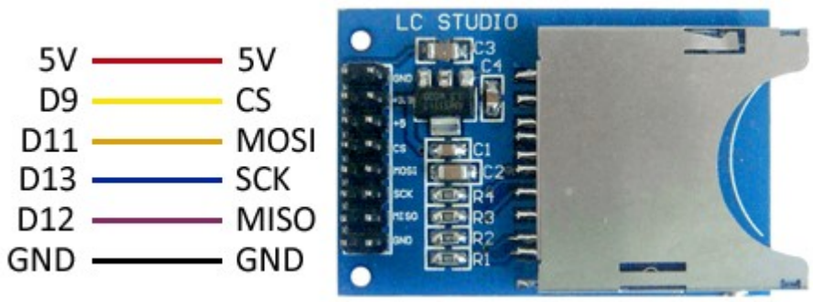


Respecto a la tarjeta, es sabido que han tenido un gran descenso en sus precios. En la actualidad, una tarjeta SD o micro SD de 1GB tiene un precio muy reducido, y es capacidad más que suficiente para la mayoría de proyectos. Incluso es frecuente que podamos reciclar antiguas tarjetas que hemos sustituido por otras de mayor capacidad.

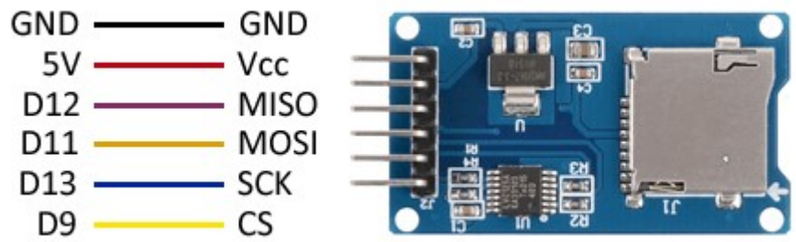
ESQUEMA DE MONTAJE

La conexión es sencilla y similar tanto para lectores SD como Micro SD. Simplemente alimentamos el módulo desde Arduino mediante 5V y Gnd. Por otro lado, conectamos los pines del bus SPI a los correspondientes de Arduino.

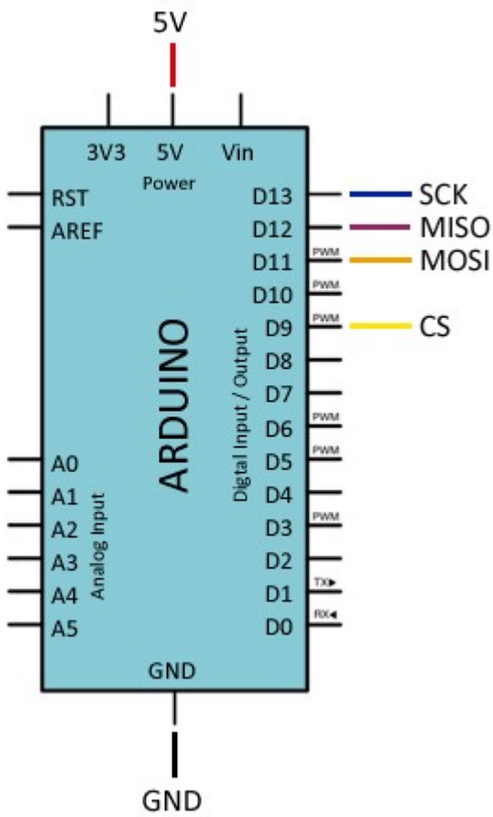
La conexión del lector SD sería la siguiente,



Similar a la de un lector micro SD, que sería la siguiente.



En ambos casos la conexión, vista desde el lado de Arduino, es la misma, y quedaría así.



Los pines SPI indicados son válidos para los modelos de Arduino Uno, Nano y Mini Pro. Para otros modelos de Arduino consultar el **esquema patillaje** correspondiente.

Verificar que vuestra placa es tolerante a un bus de 5V antes de conectarla a Arduino. Si no, tendréis que usar un adaptador de nivel lógico.

EJEMPLO DE CÓDIGO

Para realizar la lectura y escritura en la SD emplearíamos la librería SD.h, incluida en el IDE estándar de Arduino.

Esta librería incorpora funciones para el manejo de ficheros y directorios, bajo el objeto SD.

También incorpora funciones para la lectura y escritura de ficheros, bajo el objeto File.

```
// Iniciar la SD
SD.begin(cspin);

// Comprobar si existe un fichero (devuelve true si existe, falso en caso contrario)
SD.exists(filename);

// Borrar un fichero
SD.remove(filename);

// Abrir un fichero
// Mode: FILE_READ para sólo lectura
//       FILE_WRITE para lectura y escritura
SD.open(filepath, mode);

// Crear un directorio
SD.mkdir(directory);

// Eliminar un directorio
SD.rmdir(dirname);
```

```
// Obtener el tamaño de un fichero
file.size()

// Comprobar si quedan bytes por leer
file.available()

// Leer un byte del fichero
file.read()

// Escribir un byte en el fichero
file.write(data)

// Escribir una variable en un fichero (en forma similar a Serial.Print)
file.print(data)

// Obtener el punto de lectura/escritura actual
file.position()

// Mover el punto de lectura/escritura actual
// Pos: Debe estar entre 0 y file.size()
file.seek(pos)

// Cerrar el fichero
file.close()
```

La librería proporciona ejemplos de código, que resulta aconsejable revisar. Los siguientes ejemplos son modificaciones a partir de los disponibles en la librería.

LECTURA DE UN FICHERO

En este ejemplo realizamos la lectura de un fichero y mostramos el contenido por puerto Serial. El fichero leído «dataFile.txt» debe existir previamente en la tarjeta.

En un problema real el fichero podría contener, por ejemplo, los ángulos que deben adoptar los servos de un robot para adoptar distintas posiciones, separados por comas.

En la lectura de cada línea se procesaría la información leída, y se emplearía para realizar las acciones oportunas.

```
#include <SD.h>

File dataFile;

void setup()
{
  Serial.begin(9600);
  Serial.print(F("Iniciando SD ..."));
  if (!SD.begin(9))
  {
    Serial.println(F("Error al iniciar"));
    return;
  }
  Serial.println(F("Iniciado correctamente"));

  // Abrir fichero y mostrar el resultado
  dataFile = SD.open("datalog.txt");
  if (dataLine)
  {
   string dataLine;
    while (dataFile.available())
    {
      dataLine = dataFile.read();
```

```
      Serial.write(dataLine);  // En un caso real se realizarían las acciones oportunas
    }
    dataFile.close();
  }
  else
  {
    Serial.println(F("Error al abrir el archivo"));
  }
}

void loop()
{
}
```

ESCRITURA DE UN FICHERO (DATALOGGER)

El siguiente ejemplo muestra las funciones de escritura. El ejemplo simula la función de un datalogger, un proyecto muy habitual que registra periódicamente la lectura de un evento.

En este caso, empleamos una función readSensor que simula el proceso de lectura de un sensor. En el ejemplo simplemente devuelve siempre 0 pero, en un caso real, esta función se sustituiría por una función que realice la lectura de un sensor o una entrada analógica.

Cada medición guarda el tiempo de toma de datos, y el valor simulado de la lectura del sensor.

```
#include <SD.h>

File logFile;

void setup()
{
  Serial.begin(9600);
  Serial.print(F("Iniciando SD ..."));
  if (!SD.begin(9))
  {
    Serial.println(F("Error al iniciar"));
    return;
  }
  Serial.println(F("Iniciado correctamente"));
}


// Funcion que simula la lectura de un sensor
int readSensor()
{
   return 0;
}

void loop()
{
  // Abrir archivo y escribir valor
  logFile = SD.open("datalog.txt", FILE_WRITE);

  if (logFile) {
        int value = readSensor;
        logFile.print("Time(ms)=");
        logFile.print(millis());
        logFile.print(", value=");
        logFile.println(value);

        logFile.close();

  }
  else {
    Serial.println("Error al abrir el archivo");
```

```
  }
  delay(500);
}
```