

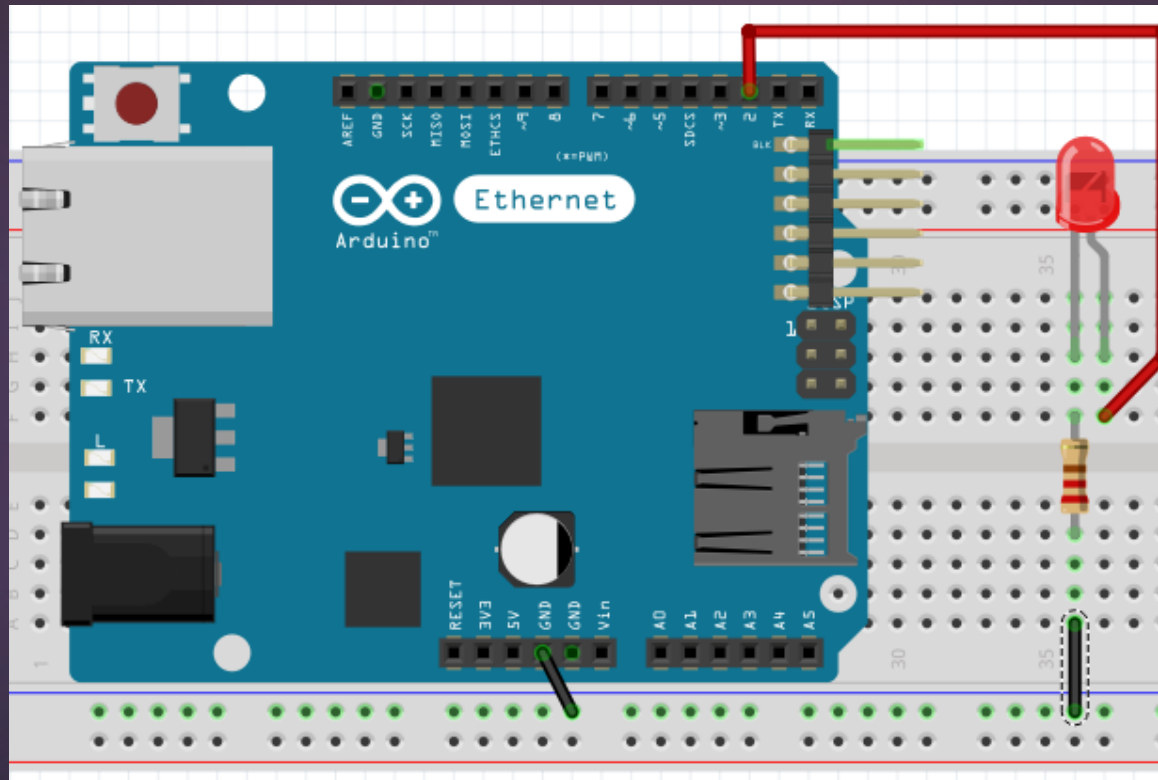
Shield Ethernet 2

TUTORA: ANGELA JAZMIN MIRANDA FLORES



EJERCICIO DE APLICACIÓN

ENCENDER Y APAGAR UN LED DESDE UNA PAGINA WEB



Tutora: Angela Jazmín Miranda Flores

Solución

```
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Dirección Física MAC
byte ip[] = { 192, 168, 1, 200 }; // IP Local que usted debe configurar
byte gateway[] = { 192, 168, 1, 1 }; // Puerta de enlace
byte subnet[] = { 255, 255, 255, 0 }; //Mascara de Sub Red
EthernetServer servidor(80); //Se usa el puerto 80 del servidor
int led=2; //led de salida
String estado="apagado";
void setup() {
  Serial.begin(9600); // Inicializa el puerto serial
  pinMode(led,OUTPUT); // Se configura como salida el pin 2
  Ethernet.begin(mac, ip, gateway, subnet); // Inicializa la conexión Ethernet y el servidor
  servidor.begin(); //inicializa el servidor
  Serial.print("El Servidor es: ");
  Serial.println(Ethernet.localIP()); // Imprime la direccion IP Local
}
```

Solución

```
void loop() {  
  // Crea una conexion Cliente  
  EthernetClient client = servidor.available();  
  if (client) {  
    boolean lineablanco=true;  
    String cadena="";  
    while (client.connected()) {  
      if (client.available()) {  
        char c = client.read();  
        cadena.concat(c);  
        int pos1=cadena.indexOf("LED=");  
        if(cadena.substring(pos1)=="LED=ENCENDER") {  
          digitalWrite(led,HIGH);  
          estado="ENCENDIDO";  
        }  
        if(cadena.substring(pos1)=="LED=APAGAR") {  
          digitalWrite(led,LOW);  
          estado="APAGADO";  
        }  
      }  
    }  
  }  
}
```

Solución

```
// si el requerimiento HTTP fue finalizado
if (c == '\n' && lineablanco) {
    client.println("HTTP/1.1 200 OK");          //envia una nueva página en código HTML
    client.println("Content-Type: text/html");  //página web HTML
    client.println();
    client.println("<!DOCTYPE HTML>");
    client.println("<html>");
    client.println("<HEAD>");
    client.println("<TITLE>ENCENDIDO Y APAGADO DE UN LED </TITLE>");
    client.println("</HEAD>");
    client.println("<BODY bgcolor: '#C5CCD3'>");
    client.println("<center>");
    client.println("<br><br>");
    client.println("<H1>Encendido y apagado de un led</H1>");
    client.println("</br></br>");
    client.println("<H2><B>estado del led:</B> </H2>");
    client.print(estado);
    client.println("");
    client.println("<br><br><br>");
    client.println("<input type=submit value=ENCENDER style=width:200px;height:75px onClick=location.href='./LED=ENCENDER\'> "); // construye el boton de encender
    client.println("<input type=submit value=APAGAR style=width:200px;height:75px onClick=location.href='./LED=APAGAR\'> "); // construye el boton de apagar
    client.println("</center>");
    client.println("</BODY>");
    client.println("</HTML>");
    break;
}
```

Solución

```
        if(c=='\n') {
            lineablanco=true;
        }else
            if(c!='\r') {
                lineablanco=false;
            }
    }
}

delay(10);
//detiene el cliente servidor
client.stop();
}
```

Encendido y apagado de un led

estado del led:

APAGADO

ENCENDER

APAGAR

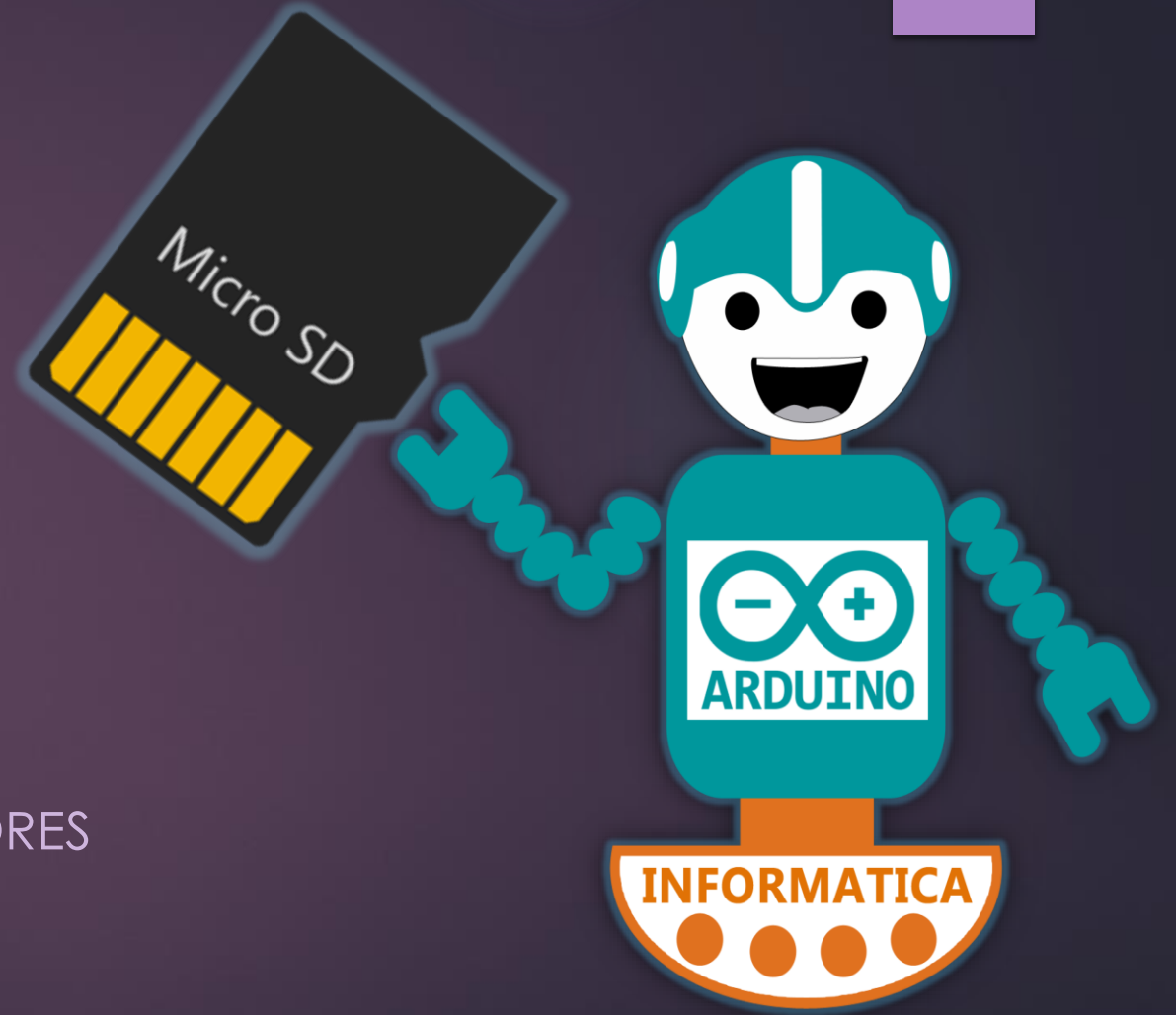
Recordando la sesión anterior

La Shield Ethernet:

- ✓ Utiliza los protocolos TCP y UDP para conectarse.
- ✓ Soporta hasta 4 conexiones simultaneas.
- ✓ Arduino se comunica con el W5100 y la tarjeta SD usando el bus SPI.
- ✓ Los pines 10,11,12,13 en Arduino UNO y los pines 50,51,52,53 en Arduino Mega no podrán utilizarse.
- ✓ Los pines 10 y 4 se usan para seleccionar el W5100 y la tarjeta SD.
- ✓ Utiliza el protocolo de IPv4.
- ✓ Solo soporta páginas Http y no Https.

Micro SD

TUTORA: ANGELA JAZMIN MIRANDA FLORES



Características

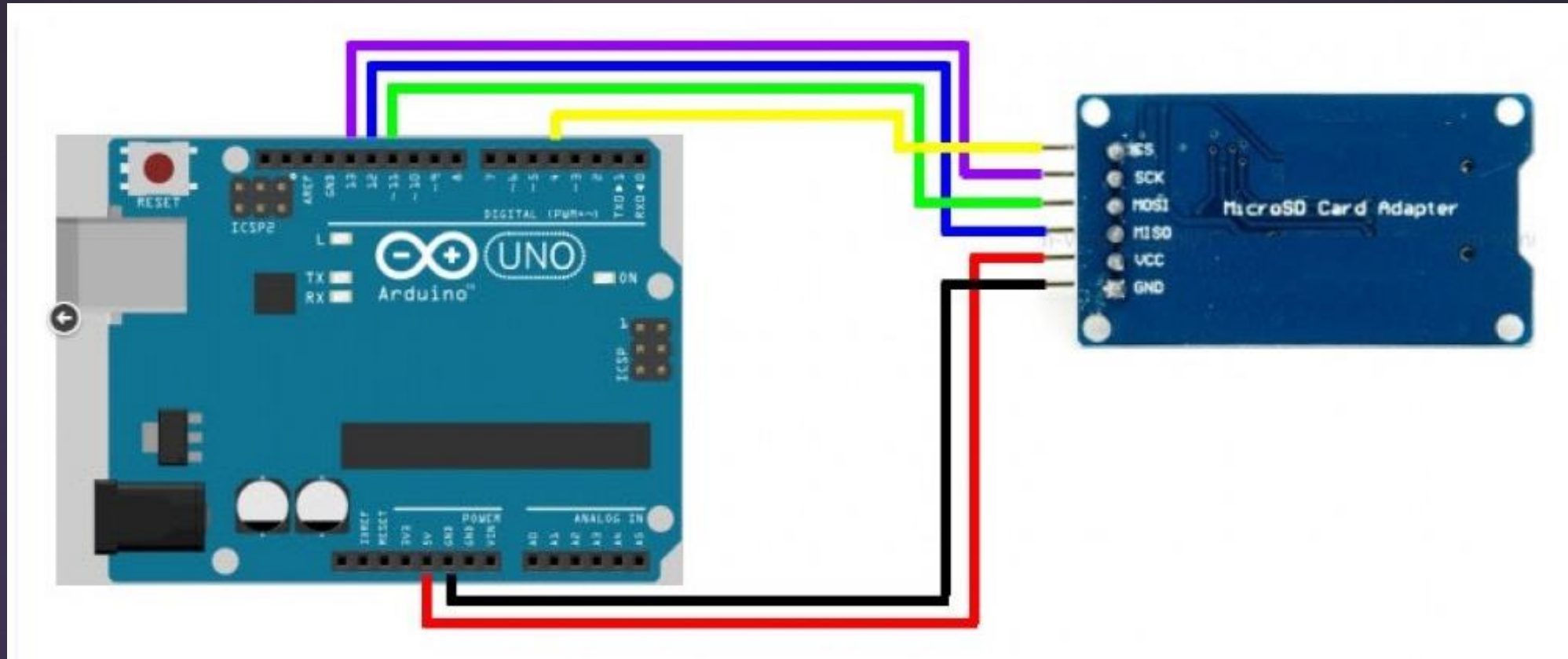
- ▶ Arduino puede leer y escribir en una tarjeta SD pero no puede formatearla.
- ▶ Solo soporta el formato FAT16 y FAT32.
- ▶ Los nombres para los archivos son limitados al viejo formato 8.3; es decir nombres de ocho caracteres y extensiones de tres.
- ▶ Utiliza su propia librería para la interacción con la tarjeta SD.

Módulo Micro SD

- ✓ Soporta tarjetas Micro SD y Micro SDHC.
- ✓ Alimentación: 4.5 a 5 V.
- ✓ Interfaz de comunicación: SPI
- ✓ Dimensiones: 45x28 mm.
- ✓ Peso: 6 gr.
- ✓ Soporta tarjetas SD hasta 4 GB.



Conexión



SPI

El protocolo SPI necesita al menos cuatro señales

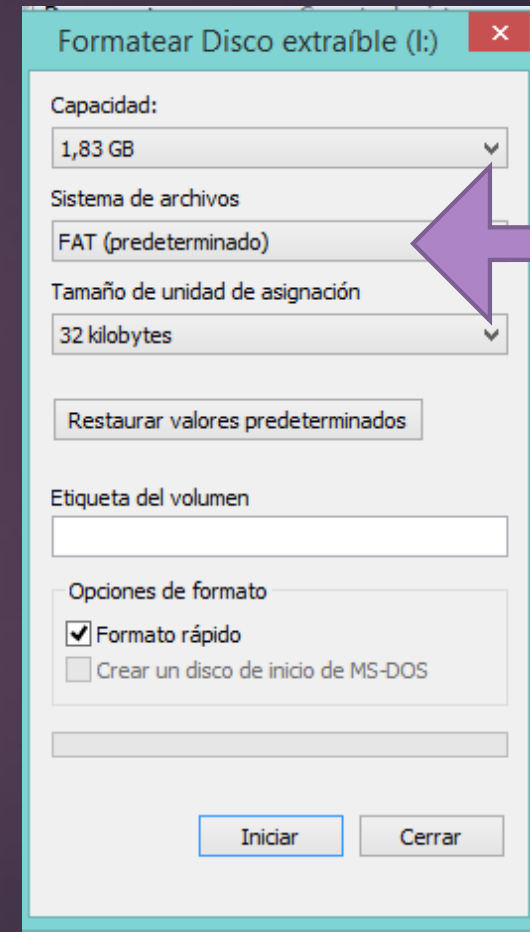
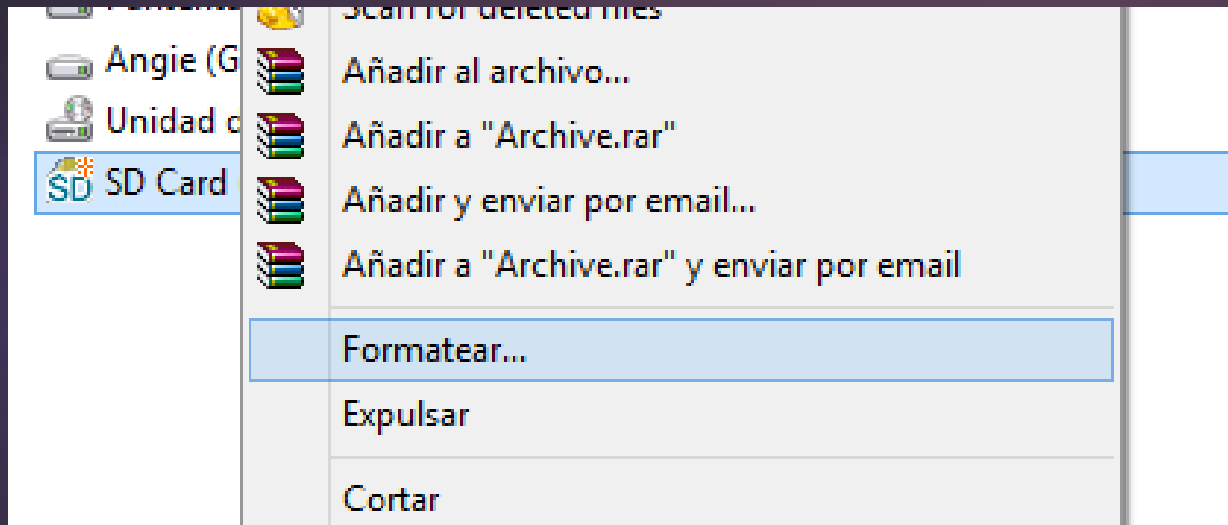
1. MOSI: Master Out Serial In (comunicación del maestro al esclavo).
2. MISO: Master In Serial Out (comunicación del esclavo al maestro).
3. SCK: Señal de reloj proporcionada por el maestro.
4. CS: The Slave Chip select (selección de la dirección del dispositivo esclavo).

Velocidad de transmisión hasta 8MHZ en arduino y Full dúplex.

El bus SPI en arduino uno se encuentra en el pin 10, en arduino mega es el pin 53.

Formateamos la Micro SD

- Le damos el formato correspondiente.



Formato de la SD

- FAT32
- FAT16

Comprobamos la tarjeta SD

```
#include <SPI.h>
#include <SD.h>

// variables de salida para la tarjeta, volumen y formato
Sd2Card card;
SdVolume volume;
SdFile root;
const int chipSelect = 4; // pin de salida de la Ethernet shield: pin 4

void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; // esperamos la conexion serial
  }
  Serial.print("\nIniciando la tarjeta SD...");
  if (!card.init(SPI_HALF_SPEED, chipSelect)) //comprobamos si la tarjeta sd esta funcionando
  {
    Serial.println("error de conexion, revisar:");
    Serial.println("* si esta la tarjeta insertada?");
    Serial.println("* si esta conectada correctamente?");
    Serial.println("* el pin de conexion es el correcto?");
    return;
  } else {
    Serial.println("tarjeta SD conectada.");
  }
}
```

Comprobamos la tarjeta SD

```
// muestra los atributos de la tarjeta SD
Serial.print("\ntipo de tarjeta: ");
switch (card.type()) {
  case SD_CARD_TYPE_SD1:
    Serial.println("SD1");
    break;
  case SD_CARD_TYPE_SD2:
    Serial.println("SD2");
    break;
  case SD_CARD_TYPE_SDHC:
    Serial.println("SDHC");
    break;
  default:
    Serial.println("desconocida");
}
if (!volume.init(card)) {/// verifica el sistema de la tarjeta SD es decir si esta en FAT16 o FAT32
  Serial.println("tarjeta en formato desconocido");
  return;
}
uint32_t volumesize;
Serial.print("\nTarjeta en formato FAT");
Serial.println(volume.fatType(), DEC);
Serial.println();
```

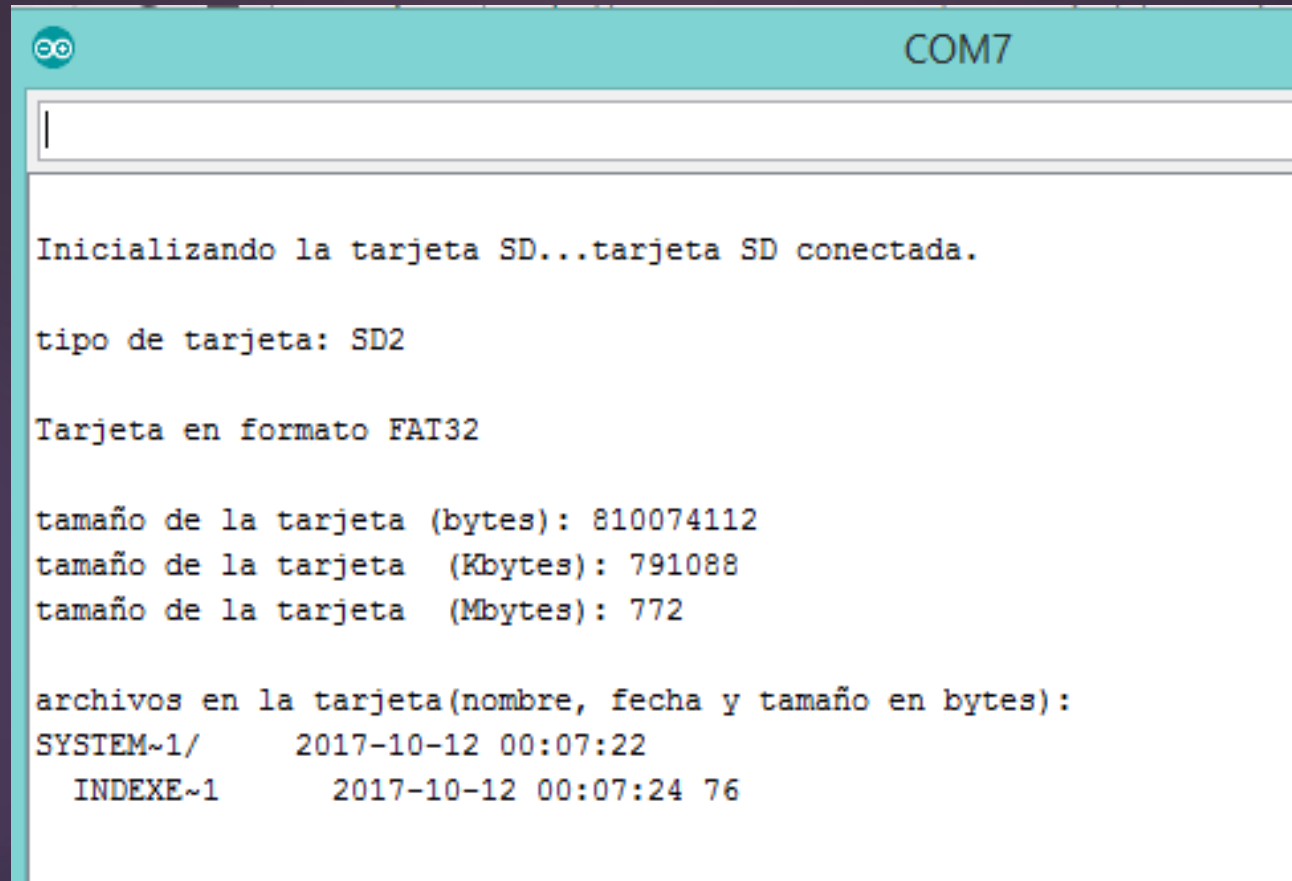

Comprobamos la tarjeta SD

```
    volumesize = volume.blocksPerCluster();
    volumesize *= volume.clusterCount();
    volumesize *= 512;                                // la tarjeta SD tiene bloques de 512 bytes
    Serial.print("tamaño de la tarjeta (bytes): ");
    Serial.println(volumesize);
    Serial.print("tamaño de la tarjeta  (Kbytes): ");
    volumesize /= 1024;
    Serial.println(volumesize);
    Serial.print("tamaño de la tarjeta  (Mbytes): ");
    volumesize /= 1024;
    Serial.println(volumesize);

    Serial.println("\narchivos en la tarjeta(nombre, fecha y tamaño en bytes): ");
    root.openRoot(volume);
    root.ls(LS_R | LS_DATE | LS_SIZE); //lista de archivos
}

void loop(void) {
  |
}
```


Salida por el monitor serial



```
COM7

Iniciando la tarjeta SD...tarjeta SD conectada.

tipo de tarjeta: SD2

Tarjeta en formato FAT32

tamaño de la tarjeta (bytes): 810074112
tamaño de la tarjeta (Kbytes): 791088
tamaño de la tarjeta (Mbytes): 772

archivos en la tarjeta(nombre, fecha y tamaño en bytes):
SYSTEM~1/      2017-10-12 00:07:22
  INDEXE~1      2017-10-12 00:07:24 76
```

Librería SD

- ▶ `SD.begin(CSpin)`: Inicializa la librería SD y la tarjeta, como parámetro se le indica el pin SS de conexión.
- ▶ `SD.exists(FILENAME)`: Comprueba si un archivo específico existe.
- ▶ `SD.mkdir(DIRECCIÓN)`: Crea el directorio especificado.
- ▶ `SD.remove(FILENAME)`: Elimina el archivo específico de la tarjeta SD.
- ▶ `SD.rmdir(DIRECCIÓN)`: Elimina el directorio especificado.
- ▶ `SD.open(FILENAME,MODULO)`: Abre el archivo especificado de acuerdo al modo que se le indique.

`FILE_READ` → solo lectura

`FILE_WRITE` → lectura y escritura

Nota: Si en la función `SD.open(FILENAME)` no mandamos el modo en el que se abrirá el archivo por defecto lo asignará como solo lectura.

Funciones de la clase FILE

- ▶ `FILE.available()`: Comprueba si hay bytes disponibles para leer en el archivo, retorna el número de bytes disponibles.
- ▶ `FILE.read()`: Lee un byte de la variable FILE.
- ▶ `FILE.write(DATO)`: Escribe un byte en el archivo.
nota: El archivo debe estar abierto en modo lectura y escritura.
- ▶ `FILE.print(DATO)`: Escribe los datos dentro el archivo.
- ▶ `FILE.size()`: Retorna el tamaño en bytes del archivo.
- ▶ `FILE.close()`: Cierra el archivo y recién se guardan los datos en la SD.

Ejercicio de aplicación

Crear un archivo “prueba.txt” en la Micro SD, agregarle datos al archivo y mostrar su contenido por el monitor serial.

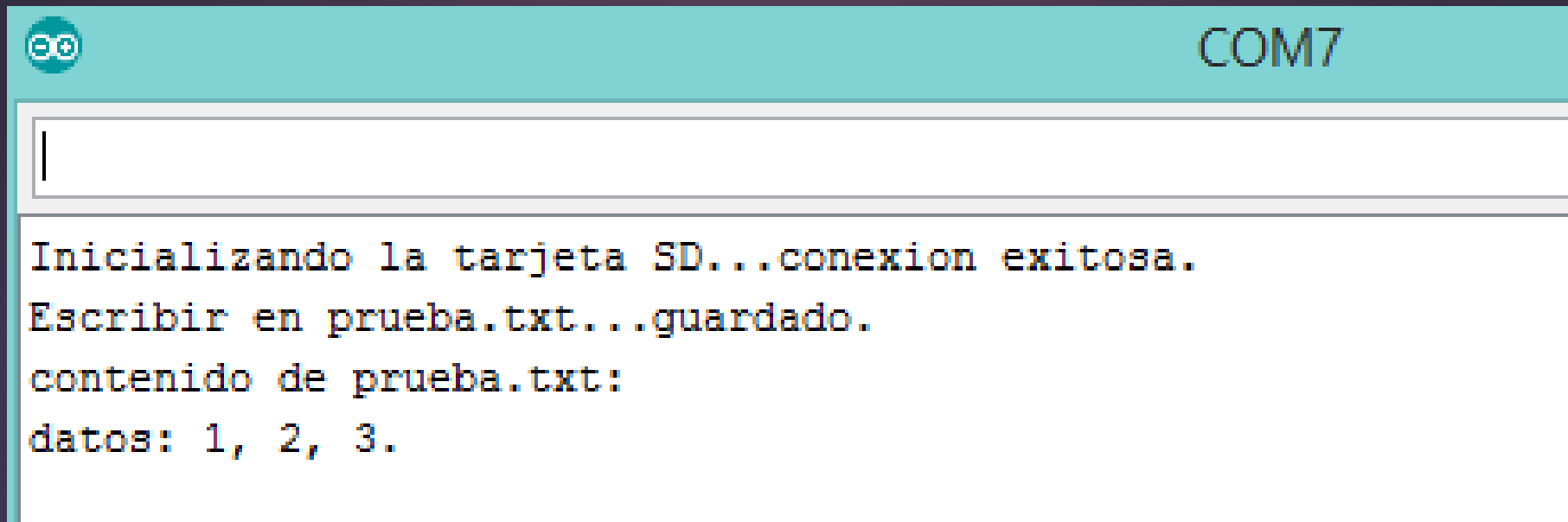
```
#include <SPI.h>
#include <SD.h>
File myFile; //definimos de tipo FILE al objeto que almacenara los archivos
void setup() {
  Serial.begin(9600);
  while (!Serial) {
    ; //esperamos a que se conecte por el puerto serial
  }
  Serial.print("Inicializando la tarjeta SD...");
  if (!SD.begin(4)) {
    Serial.println("Error en conexion");
    return;
  }
  Serial.println("conexion exitosa.");
  myFile = SD.open("prueba.txt", FILE_WRITE); //abrimos el archivo
  if (myFile) { //si el archivo ha sido abierto
    Serial.print("Escribir en prueba.txt...");
    myFile.println("datos: 1, 2, 3."); //escribe los datos dentro el archivo
    myFile.close(); //cerramos el archivo
    Serial.println("guardado.");
  } else {
    Serial.println("error al abrir prueba.txt"); //si el archivo no ha sido abierto correctamente
  }
}
```

Ejercicio de aplicación

```
//para la lectura del archivo
myFile = SD.open("prueba.txt");
if (myFile) {
    Serial.println("contenido de prueba.txt:");
    while (myFile.available()) {
        Serial.write(myFile.read());
    }
    myFile.close();//cierra el archivo
} else {
    Serial.println("error al abrir prueba.txt");
}

void loop() {
}
```

Salida por Monitor Serial



```
COM7  
|  
Iniciando la tarjeta SD...conexion exitosa.  
Escribir en prueba.txt...guardado.  
contenido de prueba.txt:  
datos: 1, 2, 3.
```

Ejercicio de aplicación

Crear un archivo llamado "sensor.txt", en el se debe almacenar el tiempo en milisegundos en el que fue leído el dato del sensor, además del dato del sensor.(nota: debe almacenarse mas de un dato dentro el archivo)

Ej:

Un sensor ultrasónico a los 10 milisegundos, detectó que un objeto se encuentra a 12 centímetros de distancia.

Tiempo (ms), dato (cm)

10 , 12

