Tutorial básico para el diseño de un Data Logger con Arduino y micro-SD

Dr. Rubén Estrada Marmolejo

Resumen.

En este tutorial se explicará el diseño de un registrador de datos (Data Logger) con **Arduino** y una memoria **micro-SD**. Los componentes que lo incluyen y el software necesario.

Palabras clave:

Arduino sd card, arduino micro sd, sd card arduino, arduino sd shield, sd card module, arduino sd, arduino data logger, micro sd arduino, sd shield arduino, data logger arduino.

Introducción

HeTPro LABs, Marcelino García Barragán 1613, Guadalajara, Jalisco, México, C.P. 44840
https://hetpro-store.com *ruben.estrada@hetpro.com.mx*

Un Data Logger es un sistema que permite registrar y guardar datos o información. Esta información puede ser variables físicas como temperatura, humedad, presión, aceleración, etc. Hasta horas de trabajo, personas que pasan por una avenida, velocidades de los vehículos que pasan por algún lugar en particular, etc. El diseño de un data logger con Arduino tiene muchas ventajas; la primera es la facilidad del diseño. Arduino es una plataforma de Hardware libre que permite a casi cualquier persona poder diseñar un sistema digital.

Figura 1. **SDlogger** registrador de Hardware Libre.
https://hetpro-store.com/sdlogger-registrador-de-datos-open-hardware/

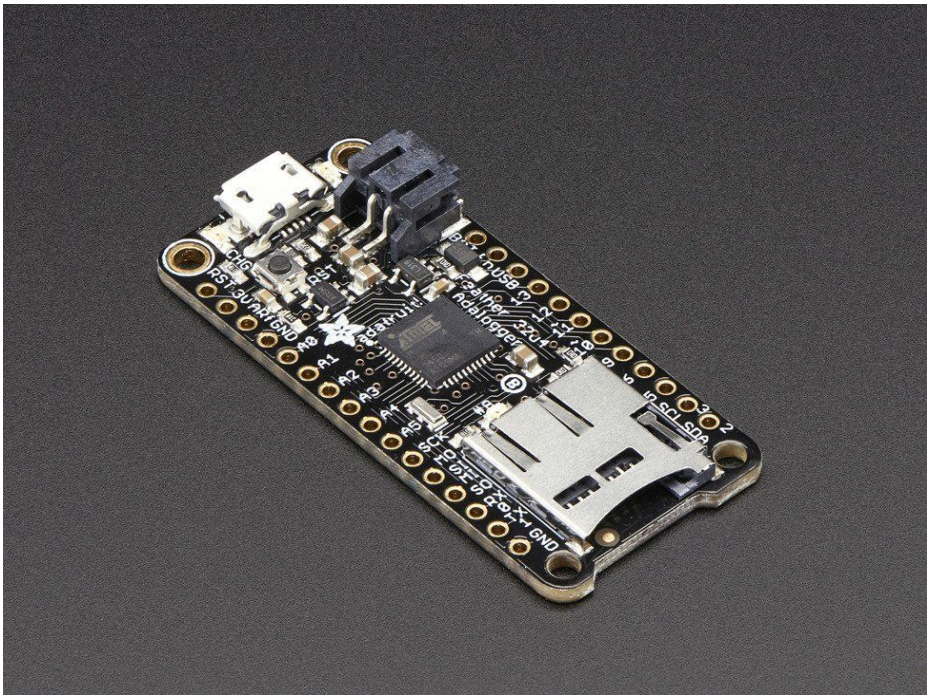


Figura 2. **Adafruit Feather 32u4** de la marca Adafruit ©.
https://hetpro-store.com/adafruit-feather-32u4-registrador-de-datos/

Un data logger diseñado con la plataforma de Arduino, requiere muy pocos elementos. Estos componentes son: Arduino, memoria micro-SD y convertidor lógico de 3.3V a 5V. La Figura 1. presenta un diseño de un Data Logger en Hardware Libre.

La Figura 2. Muestra el diseño de un Data Logger de la compañía Adafruit. La ventaja principal de este diseño es su tamaño y además cuenta con un cargador de baterías LiPo. Este aditamento le permite alimentarse de una batería. Incluye un procesador muy similar al usado por Arduino.

Elementos de un Data Logger

Los componentes principales en un registrador de datos son: el procesador, la memoria (externa y/o interna), reloj de tiempo real y el sensor. El objetivo principal de un registrador de datos es el de guardar la información en un lapso de tiempo conocido.

El tipo de **procesador** le permitirá guardar información de forma más rápida y con una mayor resolución (ADC). El procesador también le permitirá funcionar con poca energía dependiendo de su arquitectura.

La **memoria** permite guardar la información a registrar. Si se tiene poca memoria y una velocidad de captura de datos muy alta, entonces sólo se podrá guardar información en un lapso muy corto. Un aspecto importante de este elemento es el tiempo que le toma al procesador guardar los datos. Este tiempo o latencia puede perjudicar a la sincronización de la información registrada. Lo recomendable es guardar en la memoria interna del procesador un buffer o paquete de datos para cada determinado tiempo, vaciar esa información a la memoria externa no volátil.

Convertidor lógico (opcional). En el caso de que el sensor y/o la memoria no compartan el mismo voltaje lógico con el procesador, se requiere una etapa adicional para convertir los voltajes lógicos.

Reloj de Tiempo Real (RTC). Puede ser un elemento externo o interno. Su objetivo principal es el de proporcionar

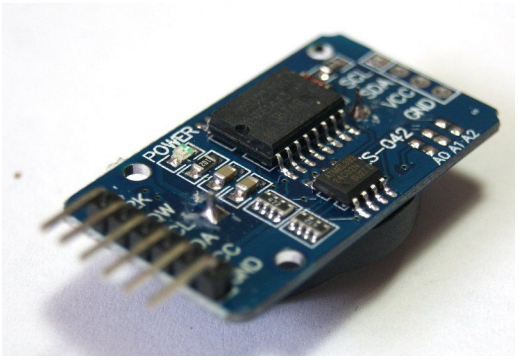
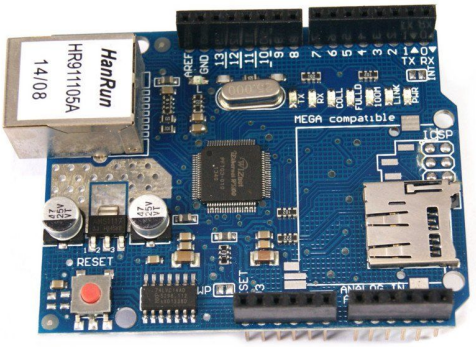
a la medición una referencia absoluta o relativa del tiempo. En que fecha, minuto, segundo, hora o año se tomo la medición.

Sensores. Los sensores son el medio por el cual podemos medir una variable. Existen principalmente dos tipos de sensores, los activos y los pasivos. Un ejemplo de un sensor pasivo es una resistencia fotosensible LDR. Esta requiere de una fuente de alimentación para poder funcionar. Un ejemplo de un sensor activo es una celda solar la cual genera un voltaje que posteriormente tendrá que amplificarse y convertirse a una señal digital.

Data logger con Arduino

Un data logger diseñado con Arduino también requiere de los elementos descritos anteriormente. La Tabla 1. muestra los componentes usados en el diseño de un registrador de datos. El procesador utilizado está basado en la tarjeta de desarrollo Arduino MEGA R3. Se optó el uso de esta tarjeta por la cantidad de memoria RAM que tiene. El RTC es el DS3231 es un reloj de tiempo real que tiene una muy buena precisión comparado con su competencia el DS1307. Se utilizó un Ethernet Shield para poder usar el conector uSD y su lógica de conversión que permite convertir los niveles lógicos de la memoria (3.3Vdc) a la lógica del Arduino 5Vdc.

Tabla 1. Componentes del diseño de un registrador de datos con Arduino MEGA R3.

Memoria externa	uSD ADATA 8GB
RTC	 DS3231
Sensor	 Sensor de corriente SCT013-100A
Convertidor lógico	 Ethernet Shield W5100 (No es un convertidor lógico) pero ya tiene esa etapa en la tarjeta y el conector a la memoria uSD.

Software de Arduino

El Arduino MEGA 2560 R3 utiliza los pines 11, 12 y 13 para la comunicación SPI. El dispositivo esclavo (micro-SD) requiere un pin adicional para poder habilitarlo (CS - Chip Select). Para poder escribir y leer datos de la memoria micro-SD se utiliza una librería de Arduino llamada SD.h. En esta sección se describe a los elementos principales de dicha librería. La librería SD soporta a los sistemas de archivo FAT16 y FAT32. La librería requiere que la memoria está formateada usando este tipo de archivos.

Ejemplo1 - Guardar el valor del ADC en una memoria micro-SD usando Arduino.

El objetivo será crear un programa que nos permita guardar el dato digital de la lectura analógica del ADC0 de Arduino. Cada 10 segundos se guardará la información del ADC así como el tiempo que le tomó guardar y procesar el dato. Este tiempo será un tiempo “relativo” utilizando a la función millis(). Relativo se refiere a que no registramos fecha/hora en específico sino el tiempo en segundos relativo al tiempo en el que lleva funcionando el arduino.

El programa tiene 4 funciones.

1. **setup( )** . Es una función que realiza lo siguiente:
    a. Configura el puerto serial del Arduino:
        i. Serial.begin(9600);

b. Configura el pin CS del SPI del Arduino.
    i. pinMode(53, OUTPUT);
c. Verifica si está conectada la memoria micro-SD
    i. if (!SD.begin(4)) { //… }

2. **registrarDato(unsigned int dato)**. Guarda un dato de 2 Bytes (unsigned int) de nombre dato en la memoria micro-SD.
    a. Registrador1 = SD.open("LOG.TXT", FILE_WRITE); . Abre y/o crea el archivo LOG.TXT.
    b. if (Registrador1) { //… } Si se pudo abrir el archivo entonces…
    c. Registrador1.print(dato); Guarda el dato en el archivo sin salto de línea.
    d. Registrador1.print(";"); Guarda un punto y coma en el archivo sin salto de línea.
    e. Registrador1.println((long int)millis()); Guarda el tiempo actual del sistema de Arduino.
    f. Registrador1.close(); Cierra el Archivo.
3. **loop( )**. Es un ciclo que se repite una y otra vez.
    a. if(salida == false) { //...} Por default esta condición siempre es verdadera por lo tanto, guarda el tiempo actual y verifica si han transcurrido el tiempo actual + 5000 milisegundos.

b. if(t0 > t1){ //… } Si ha transcurrido 5000 milisegundos, manda llamar a la función para registrar un dato y actualiza el tiempo siguiente.

4. **serialEvent( )**. Permite recibir un byte serial de forma asíncrona. Este byte se utiliza para detener el guardado de datos o continuar con el guardado de datos. Si la ventana serial de Arduino se configura con una velocidad de 9600 BAUDios y con la opción “No Line Ending”, al momento de escribir la letra A mayúscula y enviarla por el puerto serial, se detendrá el registro de los datos. En cambio si se envía un letra B mayúscula se reiniciará el proceso normal.

Descargar archivo AQUI

Para descargar el código fuente puedes darle click a nuestra página de github.

Código 1. Muestra la sección inicial y función setup del código de Arduino para un data logger.

ArduinoDataLogger §

```
void registrarDato(unsigned int dato){
    Registrador1 = SD.open("LOG.TXT", FILE_WRITE);  //Abre y/o crea el archivo LOG.TXT

  // Si se abre el archivo escribir datos.
  if (Registrador1) {
      Registrador1.print(dato);
      Registrador1.print(";");
      Registrador1.println((long int)millis());
      Registrador1.close();
      Serial.print("Se han guardado: ");
      Serial.print(cuenta++);
      Serial.println(" datos");
  }
  else{
    Serial.println("Error no se puede abrir el archivo LOG.TXT");
  }


}

void loop()
{
        if(salida == false) {
          t0 = millis();

          if(t0 > t1){
             registrarDato(analogRead(A0));
             t1=millis()+5000;
          }
        }
}
```

Código 1 - sección 2. Muestra la segunda sección y la función registrador y loop del código de Arduino para un data logger.

ArduinoDataLogger §

```
void serialEvent() {
  if (Serial.available() > 0) {
    //Leer un nuevo Byte serial
    char datoSerialByte = (char)Serial.read();
    if(datoSerialByte == 'A') {
        salida = true;
        Serial.println("Es seguro retirar la memoria");
    }
    else if(datoSerialByte == 'B'){
        salida = false;
    }
  }
}
```

Código 1 - sección 3 Muestra la tercer sección y la función serialEvent del código de Arduino para un data logger.

Formatear FAT16 una memoria micro-SD para Arduino

Este breve tutorial explica como realizar el formateo de una memoria micro-SD para soportar el sistema de archivos FAT16. Este sistema de archivos es el que requiere la librería SD.h de Arduino. El tutorial explica como hacerlo usando Linux Ubuntu 16.04 y Gparted.

Linux Ubuntu 16.04

Paso-1. Instalar gparted.

```
sudo apt-get install gparted
```

Paso-2. Verificar donde está montado la memoria micro-SD.

```
sudo fdisk -l
```

Para las pruebas que se hicieron, solamente se podrá utilizar 2Gb de espacio de la memoria micro-SD. Si no se tiene una memoria de 2Gb, se puede utilizar una memoria más grande pero sólamente 2GB. Para el siguiente paso, se considera una memoria de 8Gb para reducirla a 2Gb.

Paso-3. Abrir el programa GPARTED. Pedirá la contraseña del superusuario. Seleccionar la memoria descrita en el paso-2. Importante, asegurarse de que sea la memoria (ver el espacio del disco) y no el disco duro, de lo contrario se perderán los datos de la computadora. Una vez

seleccionada la partición de la micro-SD borrarla. Esto se hace, seleccionando click derecho sobre la partición y la opción Borrar (Delete).

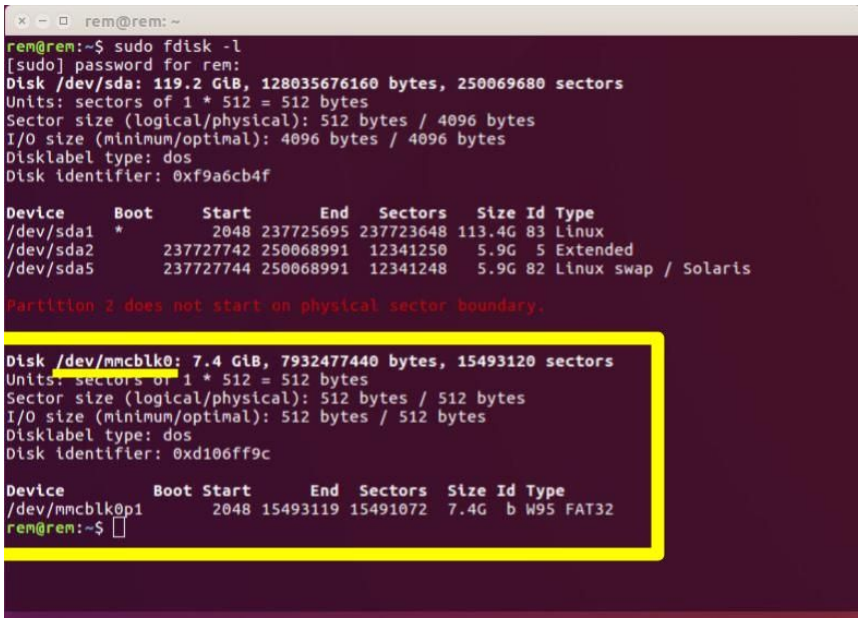


Figura 3. Muestra el comando “sudo fdisk -l” en la terminal de ubuntu 16.04. En amarillo la detección de la memoria externa uSD y su nombre: “/dev/mmcblk0”.

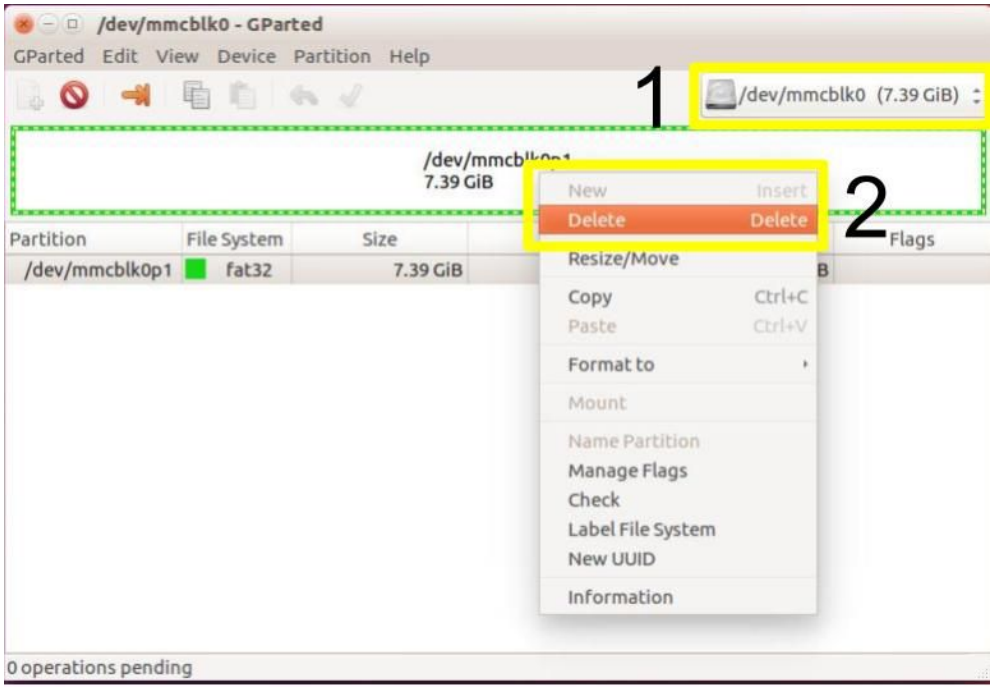


Figura 4. Muestra cómo borrar la partición de la memoria micro-SD para crear una partición en formato FAT16.

Paso-4. En esta opción la memoria aparecerá sin ninguna partición “unallocated”. Ahora crearemos una partición de 2000Mb (2Gb). Después de muchas pruebas formateando en 4Gb, 2048Mb y 3500Mb, sólamente funcionó la partición de 2000Mb. Se supone que la biblioteca SD.h también funciona con memorias de 4Gb. Cuando se utilizó una memoria de 8Gb con una partición de 4Gb no funcione correctamente. La mejor opción fue 2000Mb. Para crear la partición, seleccionar sobre la zona gris “unallocated” click derecho y el menú NEW o NUEVA Partición. En la opción de New Size (MiB) escribir 2000. En la opción File system seleccionar FAT16 y en Label ó etiqueta el nombre de la memoria. FInalmente se selecciona la opción AGREGAR o ADD.

Paso-5. Seleccionar la opción del check verde. Esto iniciará el proceso de crear la partición. Cuando termine seleccione la opción cerrar o close.

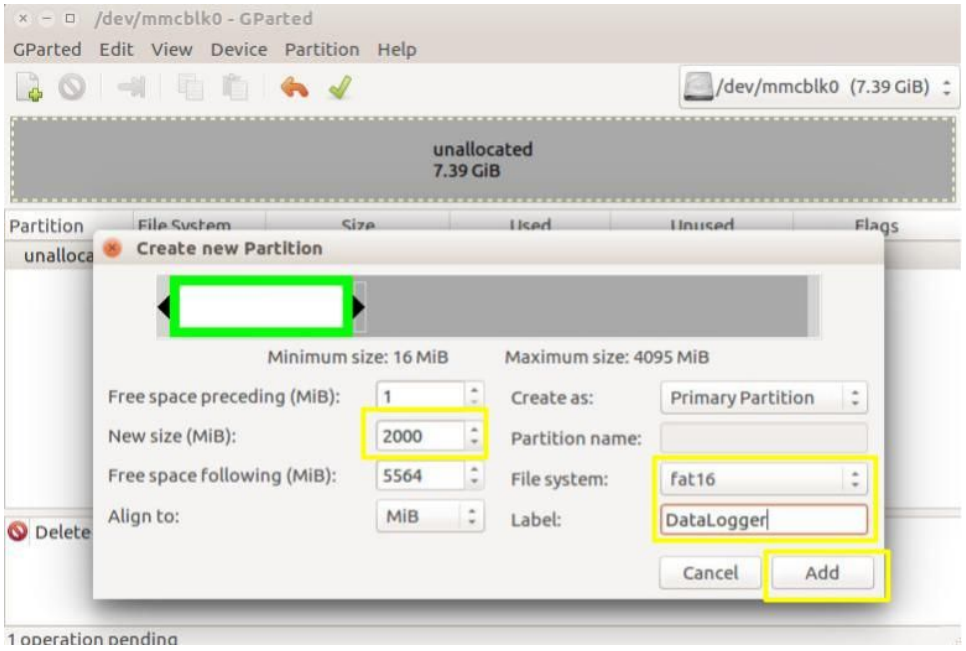


Figura 5. Muestra los pasos para crear una partición de 2000Mb, FAT16 de nombre DataLogger usando GPARTED en Ubuntu 16.04.

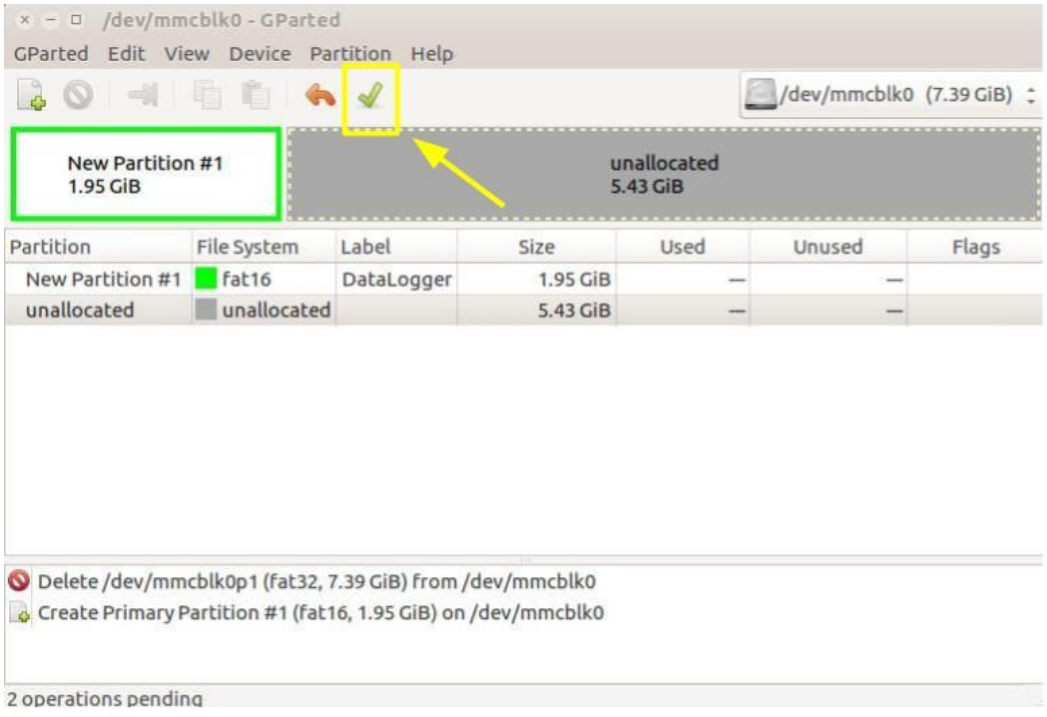


Figura 6. Muestra el último paso para finalizar el proceso de crear la partición FAT16.