MÓDULO ADAPTADOR DE MEMORIA MICRO SD CON PINES P/ARM AVR

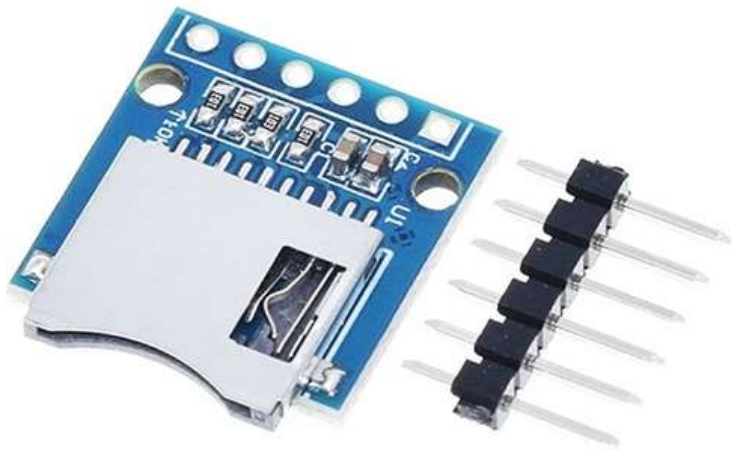
OKY3002-2

OKYSTAR

AG Electrónica

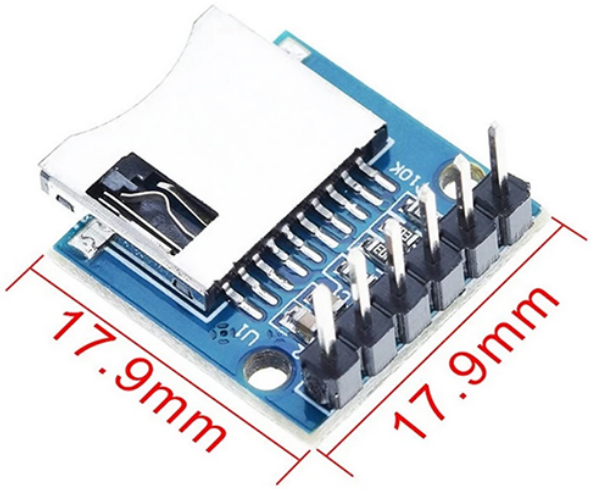
MÓDULO ADAPTADOR DE MEMORIA MICRO SD CON PINES P/ARM AVR

OKY3002-2



DESCRIPCIÓN

El módulo adaptador de memoria micro SD permite emplear como almacenamiento una tarjeta SD, que se puede incorporar a proyectos de electrónica. Solución sencilla para la transferencia de datos hacia y desde una tarjeta microSD estándar. El pinout es directamente compatible con Arduino, pero también se puede utilizar con otros microcontroladores.



CARACTERÍSTICAS

- Autobús SPI
- Operación de 3.3V

Las tarjetas SD se utilizan comúnmente para aplicaciones como el registro de temperatura, donde la hora y la temperatura se registran durante largos períodos de tiempo. También se puede utilizar para servir grandes cantidades de datos, como imágenes gráficas.

Este módulo de tarjeta MicroSD debe considerarse como una ranura para tarjetas sin formato y funciona a 3.3 V, que es el voltaje al que funciona la tarjeta MicroSD. Las líneas de datos deben estar en niveles lógicos de 3.3 V para evitar daños a la tarjeta MicroSD, por lo que este módulo se utiliza mejor con MCU de 3.3V. Si se usa con una MCU de 5V, se requieren cambiadores de nivel lógico externos para las líneas del bus SPI para evitar daños.

Las líneas de datos tienen resistencias pull-up de 10K a 3.3V en el módulo.

El consumo de corriente de 3.3 V dependerá de la tarjeta que se utilice con el módulo. Cuando está inactiva, la tarjeta puede consumir 500uA. Al leer la tarjeta, lo común es 15-30 mA. Las operaciones de escritura consumen más corriente y se informa que algunas tarjetas requieren hasta 100 mA para las operaciones de escritura, por lo que esto debe tenerse en cuenta al seleccionar la fuente de alimentación. La alimentación Arduino de 3.3 V, que suele ser buena para aproximadamente 30-50 mA, normalmente funcionará bien para leer la tarjeta, pero puede ser insuficiente para operaciones de escritura. Si tiene problemas de escritura, esto es lo primero que debe comprobar.

PINOUT

El módulo tiene instalado un cabezal macho, el módulo se puede insertar directamente en una placa, lo cual es útil ya que el etiquetado de los pines es visible o se pueden usar puentes hembra estilo Dupont para realizar conexiones.

- 3V3 = potencia de 3.3V
- CS = Selección de chip SPI
- MOSI = SPI MOSI, se conecta a MOSI en MCU
- CLK = Reloj SPI
- MISO = SPI MISO, se conecta a MISO en MCU
- GND = Tierra, debe ser común con la MCU

CÓDIGO DE EJEMPLO

```
// incluye la biblioteca SD:
#include  < SPI . h >
#incluir  < SD . h >

// configurar variables utilizando las funciones de la biblioteca
de la utilidad SD:
tarjeta Sd2Card  ; Volumen SdVolume ; Raíz del archivo SD ;

// cambia esto para que coincida con tu módulo o escudo SD;
const  int  chipSelect  =  53 ;

 configuración nula ( )  {
  Serial . comenzar ( 9600 ) ;
  }

  De serie . print ( "\nInicializando tarjeta SD..." ) ;

  // usaremos el código de inicialización de las bibliotecas de
utilidades
  // ya que solo estamos probando si la tarjeta funciona.
  if  ( ! card . init ( SPI_HALF_SPEED ,  chipSelect ) )  {
    Serial . println ( "falló la inicialización. Cosas que
comprobar:" ) ;
    De serie . println ( "*¿Hay una tarjeta insertada?" ) ;
    De serie . println ( "*¿Su cableado es correcto?" ) ;
    De serie . println ( "* ¿Cambiaste el pin chipSelect para que
coincida con tu escudo o módulo?" ) ;
    mientras  ( 1 ) ;
  }  más  {
    Serie . println ( "El cableado es correcto y hay una tarjeta
presente." ) ;
  }

  //imprime el tipo de tarjeta
  Serial . imprimirln ( ) ;
  De serie . print ( "Tipo de tarjeta: " ) ;
  switch  ( card.type ( ) ) { case SD_CARD_TYPE_SD1 : Serial . _ _
imprimirln ( "SD1" ) ; romper ; caso SD_CARD_TYPE_SD2 : Serie .
imprimirln ( "SD2" ) ; romper ; caso SD_CARD_TYPE_SDHC : Serie .
println ( "SDHC" ) ; romper ; predeterminado : Serie . println
( "Desconocido" ) ; }

// Ahora intentaremos abrir el 'volumen'/'partición'; debería ser
FAT16 o FAT32
 if  ( ! volumen . init ( tarjeta ) )  {
```

```
    Serial . println ( "No se pudo encontrar la partición
FAT16/FAT32.\nAsegúrese de haber formateado la tarjeta" ) ;

    mientras  ( 1 ) ;

  }

  De serie . imprimir ( "Clústeres: " ) ;

  De serie . println ( volumen.clusterCount ( ) ) ; _ _ De serie . print
( "Bloques x Clúster: " ) ; De serie . println ( volumen.blocksPerCluster
( ) ) ; _ _

  De serie . print ( "Total de bloques: " ) ;

  De serie . println ( volumen.blocksPerCluster ( ) *
volumen.clusterCount ( ) ) ; _ _ _ _ De serie . imprimirln ( ) ;

  // imprime el tipo y tamaño del primer volumen tipo FAT

  uint32_t  volumesize ;

  De serie . print ( "El tipo de volumen es: FAT" ) ;

  De serie . println ( volumen.fatType ( ) , DEC ) ; _ _

  tamaño del volumen  =  volumen . bloquesPorCluster ( ) ;     // los
clústeres son colecciones de bloques

  tamaño de volumen  *=  volumen . clusterCount ( ) ;        // tendremos
muchos clusters

  volumesize  /=  2 ;                              // Los bloques de la
tarjeta SD son siempre de 512 bytes (2 bloques son de 1 KB)

  Serial . print ( "Tamaño del volumen (Kb):" ) ;

  De serie . println ( tamaño del volumen ) ;

  De serie . print ( "Tamaño del volumen (Mb): " ) ;

  tamaño del volumen  /=  1024 ;

  De serie . println ( tamaño del volumen ) ;

  De serie . print ( "Tamaño del volumen (Gb): " ) ;

  De serie . println ( ( flotante ) tamaño volumen  /  1024.0 ) ;

  De serie . println ( "\nArchivos encontrados en la tarjeta (nombre,
fecha y tamaño en bytes): " ) ;

  raíz . openRoot ( volumen ) ;

  // enumera todos los archivos en la tarjeta con fecha y tamaño

  raíz . ls ( LS_R  |  LS_FECHA  |  LS_TAMAÑO ) ;

}

 bucle vacío ( vacío )  {

}
```

REALIZÓ: OACH

REVISÓ: GAC