Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

SD Module (SKU: DFR0071)

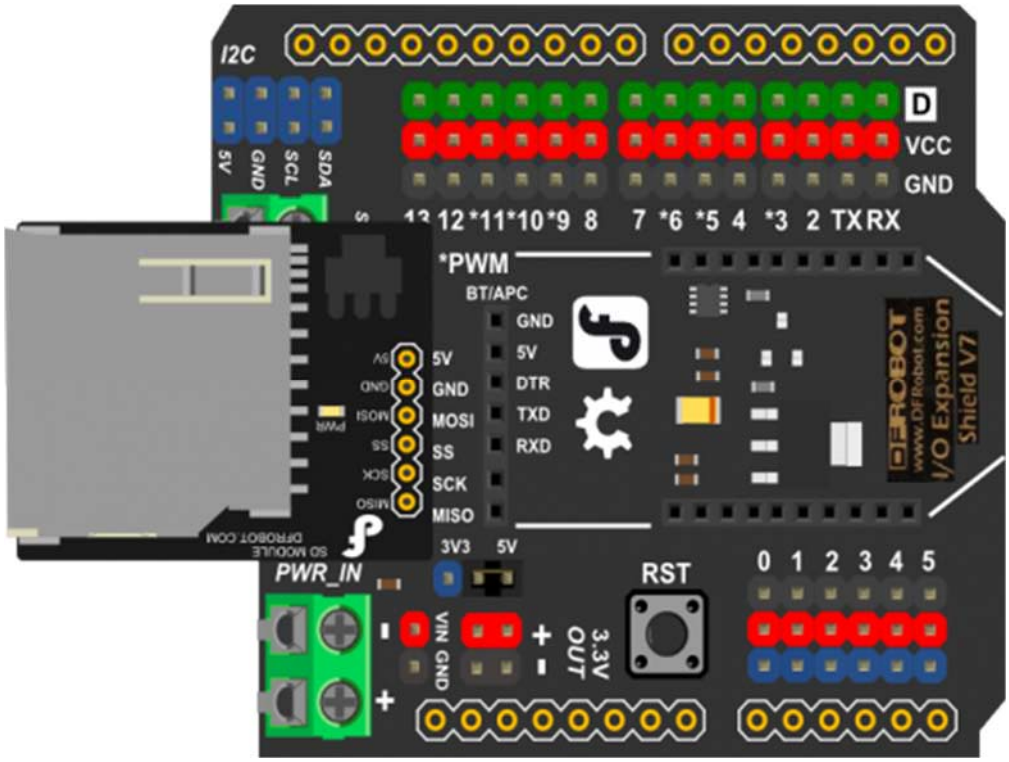
Contents

Introduction

The DFRobot SD Card Shield is a simple solution for transferring data to and from a standard SD card. The pinout is directly compatible with Arduino, but can also be used with other microcontrollers. It allows you to add mass storage and data logging to your project. It sits directly on a Arduino and contains a switch to select the flash card slot.

Connection Diagram

Insert arduino expansion shield V7 onto UNO. Insert SD Module on the arduino expansion shield V7 SD(SPI) socket:



Features

- Break out board for standard SD card and Micro SD (TF) card
- Sits directly on a Arduino
- Also be used with other microcontrollers

Sample Code

Open the sample sketch "SD\examples\CardInfo".

```
/*
  SD card test


 This example shows how use the utility libraries on which the'
 SD library is based in order to get info about your SD card.
 Very useful for testing a card when you're not sure whether its working or n
ot.
```

```
 The circuit:
  * SD card attached to SPI bus as follows:
 ** MOSI - pin 11 on Arduino Uno/Duemilanove/Diecimila
 ** MISO - pin 12 on Arduino Uno/Duemilanove/Diecimila
 ** CLK - pin 13 on Arduino Uno/Duemilanove/Diecimila
 ** CS - depends on your SD card shield or module.
               Pin 4 used here for consistency with other Arduino examples


 created  28 Mar 2011
 by Limor Fried
 modified 9 Apr 2012
 by Tom Igoe
 */
 // include the SD library:
#include <SD.h>

// set up variables using the SD utility library functions:
Sd2Card card;
SdVolume volume;
SdFile root;

// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
const int chipSelect = 4;

void setup()
{
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
```

```
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  Serial.print("\nInitializing SD card...");
  // On the Ethernet Shield, CS is pin 4. It's set as an output by default.
  // Note that even if it's not used as the CS pin, the hardware SS pin
  // (10 on most Arduino boards, 53 on the Mega) must be left as an output
  // or the SD library functions will not work.
  pinMode(10, OUTPUT);     // change this to 53 on a mega


  // we'll use the initialization code from the utility libraries
  // since we're just testing if the card is working!
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card is inserted?");
    Serial.println("* Is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your shield
or module?");
    return;
  } else {
   Serial.println("Wiring is correct and a card is present.");
  }


  // print the type of card
  Serial.print("\nCard type: ");
  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("Unknown");
  }


  // Now we will try to open the 'volume'/'partition' - it should be FAT16 or
FAT32
  if (!volume.init(card)) {
    Serial.println("Could not find FAT16/FAT32 partition.\nMake sure you've f
ormatted the card");
    return;
  }



  // print the type and size of the first FAT-type volume
  uint32_t volumesize;
  Serial.print("\nVolume type is FAT");
  Serial.println(volume.fatType(), DEC);
  Serial.println();


  volumesize = volume.blocksPerCluster();    // clusters are collections of b
locks
  volumesize *= volume.clusterCount();       // we'll have a lot of clusters
  volumesize *= 512;                            // SD card blocks are always
512 bytes
  Serial.print("Volume size (bytes): ");
  Serial.println(volumesize);
  Serial.print("Volume size (Kbytes): ");
  volumesize /= 1024;
  Serial.println(volumesize);
  Serial.print("Volume size (Mbytes): ");
  volumesize /= 1024;
  Serial.println(volumesize);
```

```
  Serial.println("\nFiles found on the card (name, date and size in bytes): "
);
  root.openRoot(volume);

  // list all files in the card with date and size
  root.ls(LS_R | LS_DATE | LS_SIZE);
}


void loop(void) {


}
```