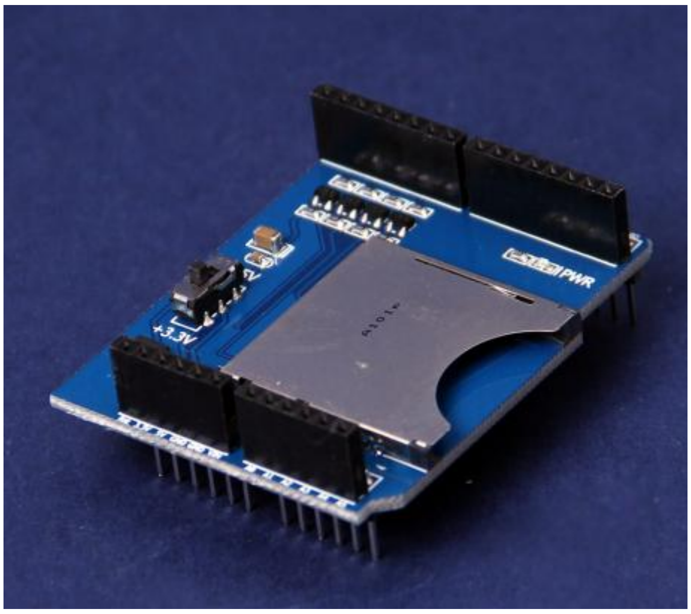
ARDUINO SHIELD MICRO SD
User Manual



Hardware Overview:

The features of the microSD Shield are as follows:

1. A large protoyping area where you can solder other parts for your project.
2. A microSD socket.
3. A power LED. Lights up as soon as the Arduino is powered.
4. A reset button connected to the Arduino's reset line.

Code Example:

Press your completed microSD Shield onto your Arduino, and connect the board to your computer with the Arduino's USB cable.

For this example, you will use the SD library that comes with every Arduino IDE installation. There are additional reference materials for the SD library on the Arduino site. The SD library example sketches are in your Arduino examples folder under Files > Examples > SD

microSD shield uses Pin 8 as the chip select line. To use the SD library with the shield, you'll need to change the line `const int chipSelect = 4;` to `const int chipSelect = 8;` in the CardInfo.ino sketch (the sketch below is already modified if you'd like to copy and paste it instead), then press Upload.

COPY CODE:

```
/*
 SD card test
 MOSI - 11
 MISO - 12
 CLK - 13
 CS - 8
*/
// include the SD library:
#include <SPI.h>
#include <SD.h>
// set up variables using the SD utility library functions:
Sd2Card card;
SdVolume volume;
SdFile root;
// The Sparkfun microSD shield uses pin 8 for CS
const int chipSelect = 8;
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...");
// 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);
// 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 formatted 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 blocks
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) {
}
```

Open your Arduino serial monitor by going to Tools > Serial Monitor to see the results of the card test.

Once your card is detected, you can move on to using that extra space for something! Below is a simple analog data logging sketch from the SD card library with the same modification- changing the CS pin to 8.`const int chipSelect = 8;`

COPY CODE

```
#include <SPI.h>
#include <SD.h>
// On the Ethernet Shield, CS is pin 4. Note that even if it's not
// used as the CS pin, the hardware CS pin (10 on most Arduino boards,
// 53 on the Mega) must be left as an output or the SD library
// functions will not work.
const int chipSelect = 8;
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("Initializing SD card...");
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  pinMode(10, OUTPUT);

  // The chipSelect pin you use should also be set to output
  pinMode(chipSelect, OUTPUT);

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
   Serial.println("Card failed, or not present");
   // don't do anything more:
   return;
  }
  Serial.println("card initialized.");
 }
 void loop()
 {
  // make a string for assembling the data to log:
  String dataString = "";
  // read three sensors and append to the string:
  for (int analogPin = 0; analogPin < 3; analogPin++) {
   int sensor = analogRead(analogPin);
   dataString += String(sensor);
   if (analogPin < 2) {
    dataString += ",";
   }
  }
```

```
// open the file. note that only one file can be open at a time,
// so you have to close this one before opening another.
File dataFile = SD.open("datalog.txt", FILE_WRITE);

// if the file is available, write to it:
if (dataFile) {
  dataFile.println(dataString);
  dataFile.close();
  // print to the serial port too:
  Serial.println(dataString);
}
// if the file isn't open, pop up an error:
else {
  Serial.println("error opening datalog.txt");
}
```