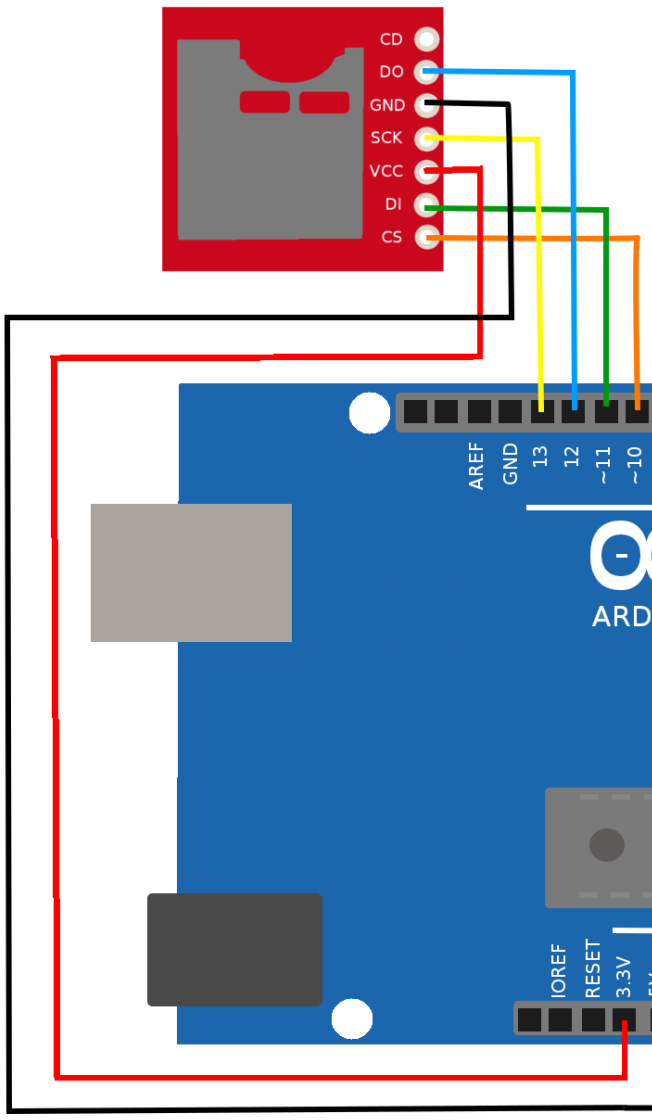
microSD Card Socket

The micro SD Card Socket provides us with an SPI connection to a microSD card. This can be used to store large amounts of data, and can be written to by the Arduino, and then read by a computer on which we view and analyze the data in innumerable ways.

Wiring

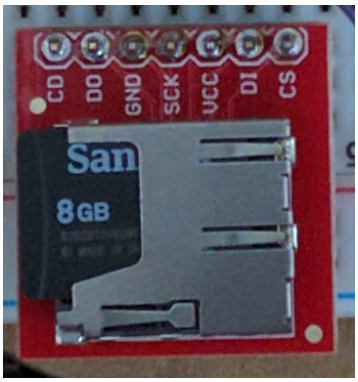
Start by wiring the device as shown in the diagram below:



	Device Pin	Arduino Pin	Name
	VCC	3.3V	Power
	GND	GND	Ground
	SCK	13	Serial Clock Line
	SDO	12	Serial Data Out
	SDI	11	Serial Data In
	CS	10	Chip Select

microSD card

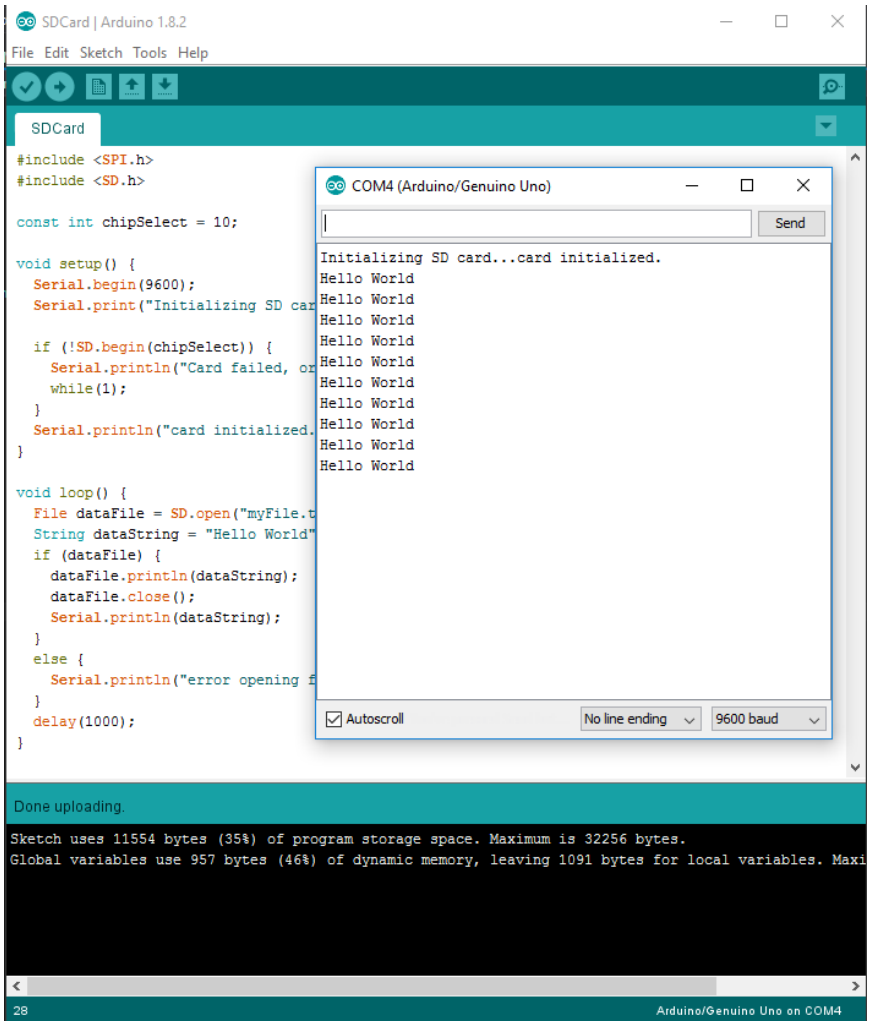
The device we have just wired does not contain any digital circuitry or processors. It's just a mechanical and electrical socket to connect to a mircoSD card. This card contains the circuitry that actually does the reading and writing of data over an SPI connection. Press the micro SD card into the socket as shown in the image below. Make sure you press it in until it clicks.



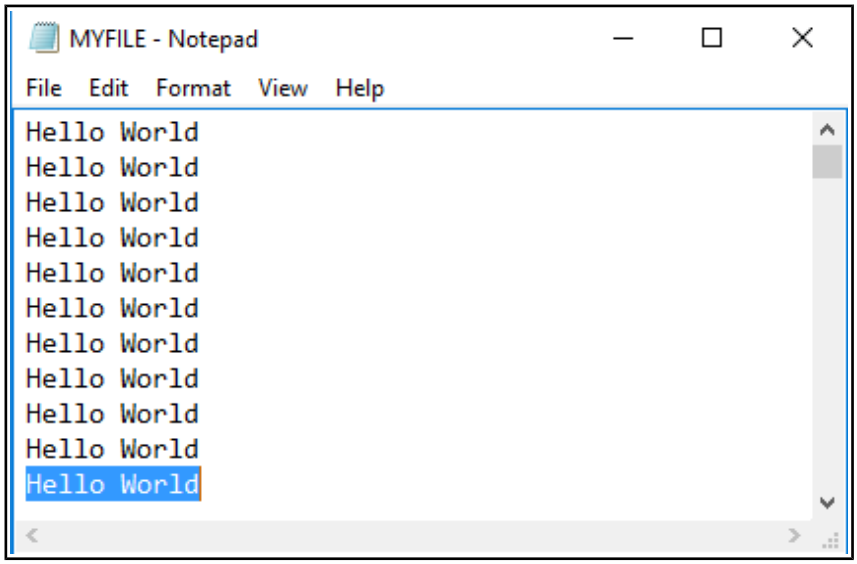
It should be noted that the SD card needs to be formatted as a FAT32 file system. This has already been done for you, so you don't need to it yourself. If you try using a different SD card, you might need to format it before it will work. A device of this format can be read by Windows, Mac, and Linux systems, so you should not need to anything special to retrieve your data from a device that is formatted this

Testing

Open the included sketch 'SDCard.ino.' Upload this sketch and you should see that, after an initialization message, the words 'Hello World' are written the serial monitor every second. This text is also being written to a text file on the SD Card. Let this run for a few moments to put a few lines of text into that file.



Next we will unpower the Arduino by disconnecting the USB cable. Now we can safely remove the micro SD card. Place the micro SD card into the included micro-->standard SD adapter. This can be placed into the SD card reader on your computer. If you are working with a computer without an SD card reader, USB SD card readers are available. Once inserted, you should see the new drive recognized by the computer, and its contents can now be explored like any other drive. You should see that a new file called 'myFile.txt' has been created.



Code walkthrough

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

Here we first include the low level SPI library that enables the Arduino to implement the SPI interface. Next we include the SD card library that uses the SPI interface to communicate with the SD card.

```
const int chipSelect = 10;
```

this specifies that the chip select line will be on pin 10. More about this in the lecture on SPI interface.

```
void setup() {
  Serial.begin(9600);
  Serial.print("Initializing SD card...");

 if (!SD.begin(chipSelect)) {
  Serial.println("Card failed, or not present");
  while(1);
  }
  Serial.println("card initialized.");
}
```

In the setup function we initialize the serial interface and print a brief message. Next, we attempt to initialize the SD card and either print a success message or print a failure message and then stop execution. We then we begin the SD card by passing it the chipSelect pin.

```
void loop() {
  File dataFile = SD.open("myFile.txt", FILE_WRITE);
  String dataString = "Hello World";
  dataFile.println(dataString);
  dataFile.close();
  Serial.println(dataString);
  delay(1000);
}
```

We start the loop function by creating a new variable of type File, and we call it data File. This is the file into which we will write our data. We call this file 'myFile.txt.' Next we create a variable of type string and call it dataString. We set the value of this string to be the text "Hello World." We write this string into the text file with println. We then close the data file to complete the operation. Finally, we print that same value to the serial port for so we can see it working; and then wait 1 second before repeating the function.