



Arduino Ethernet + SD Card

Created by Mike Barela

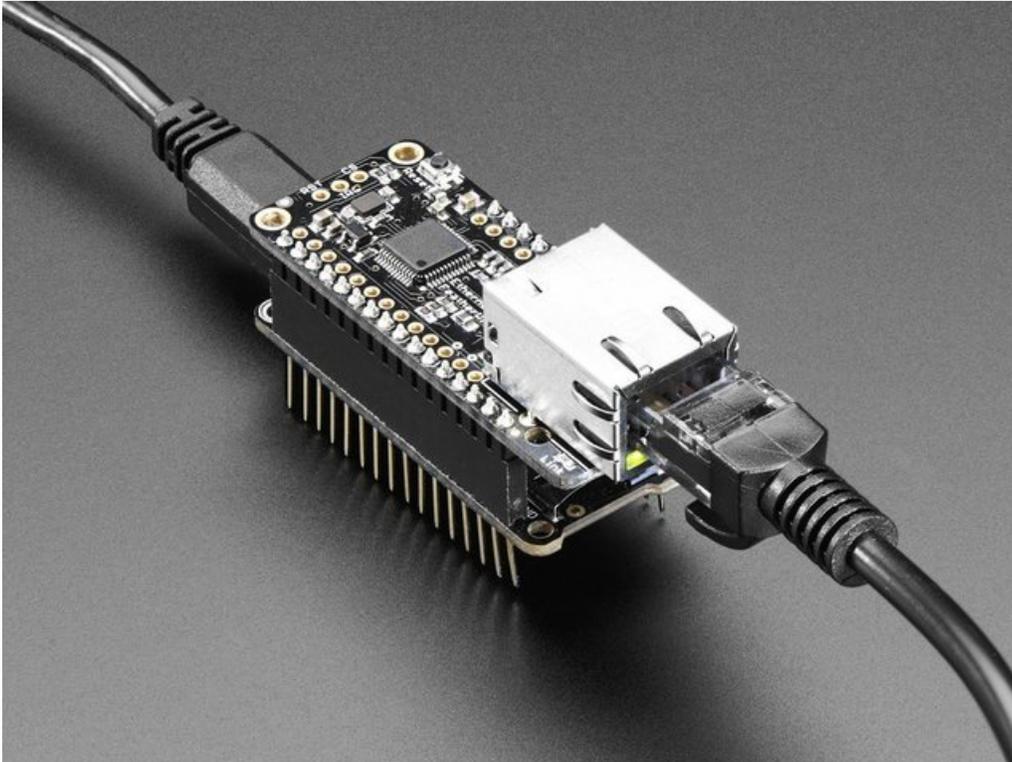


Last updated on 2018-08-22 04:06:59 PM UTC

Guide Contents

| | |
|-----------------------------|----|
| Guide Contents | 2 |
| Overview | 3 |
| SD Card Preparation | 5 |
| Reading SD Card Information | 7 |
| Listing Files | 10 |
| Serving Files over Ethernet | 12 |
| How it Works and Wrap-up | 18 |
| Wrap-Up | 19 |

Overview



This tutorial provides the basics for using a WIZ5500-based Ethernet Shield with an Arduino compatible. The Wiznet WIZ5500 is a modern Ethernet interface chip and is used in the [Ethernet Shield W5500 \(https://adafru.it/BQY\)](https://adafru.it/BQY) and the [Feather Ethernet Wing \(https://adafru.it/wiB\)](https://adafru.it/wiB) available at Adafruit.

The shield form factor works well for ATmega328 based Arduino compatibles like the Arduino Uno R3 and the [Adafruit Metro 328 \(https://adafru.it/METROXMETR\)](https://adafru.it/METROXMETR) (Classic). The shield contains both an Ethernet connection for networks and an SD Card slot for storing data.

The Feather wing works well with a Feather with a SD card slot. The [Feather 32u4 Adalogger \(https://adafru.it/tUd\)](https://adafru.it/tUd) has an Arduino Leonardo compatible 32u4 processor and a micro-SD card slot. With the Ethernet Feather Wing, it provides the same functionality as the Ethernet Shield+Metro/Uno in a much smaller package. SD and micro-SD are the same for all purposes but the latter is smaller. In this tutorial, when it says do something to an SD card and you are using the Feathers, think micro-SD.

This tutorial assumes you already know the basics of the Arduino IDE, code generation and the Arduino interface. If you are not so familiar with the Arduino IDE, you might check the tutorials in the [Adafruit Learn Arduino Series \(https://adafru.it/BQZ\)](https://adafru.it/BQZ).

As of the date of this revised tutorial, [Arduino notes their Ethernet Shields are retired \(https://adafru.it/BQ-\)](https://adafru.it/BQ-). If you use a retired or third-party Ethernet shield, you may have to use a different Arduino library that supports the chipset the board uses. Note that the Ethernet Shield and Feather Wing Adafruit uses is based on the WIZ5500 chip, not the older WIZ5100 chipset or others. The WIZ5500 requires the Ethernet2 library, not the older Ethernet library on Arduino. If you use the older hardware, just be sure you change libraries back from Ethernet2 to Ethernet and double check things.

This tutorial will go through preparing then using the SD card, both in general then reading the files on a card. The final example will show how to access the SD card remotely over Ethernet. This capability could be the basis for a remote

file storage or other program that access a remote Arduino compatible.

SD Card Preparation



WARNING: DO NOT FORMAT YOUR HARD DISK(S) TRYING TO FORMAT YOUR SD CARD.

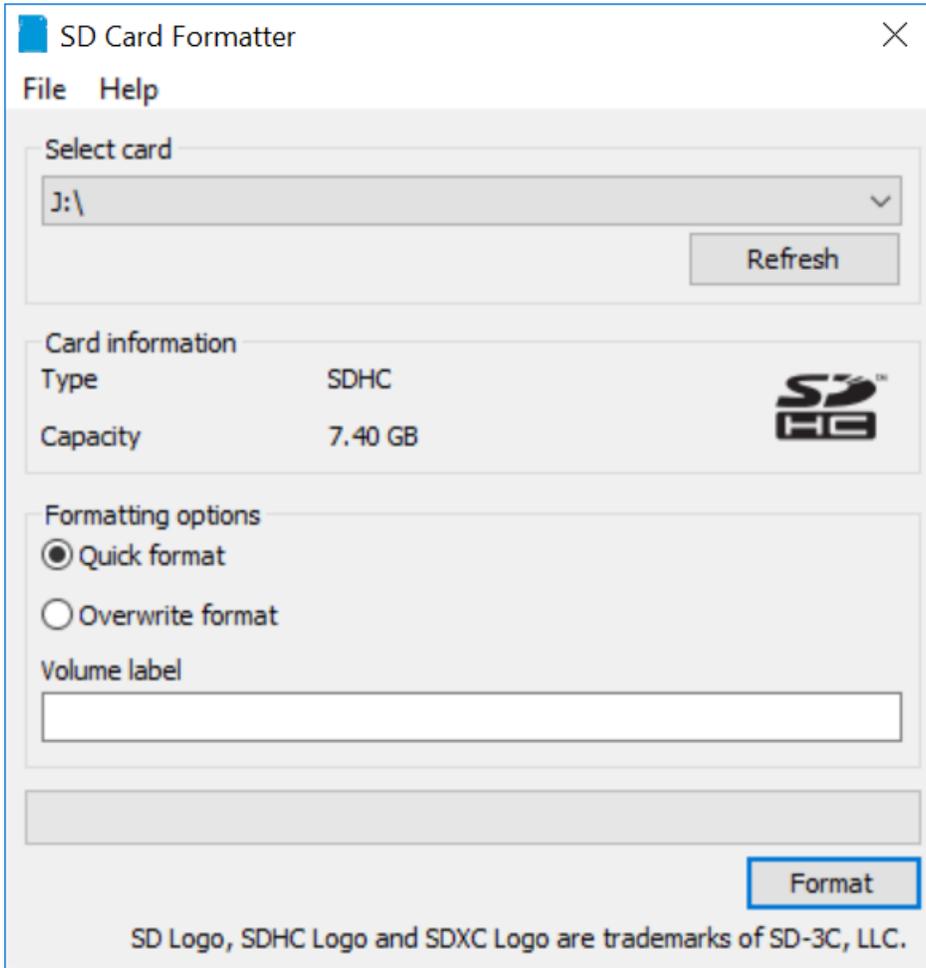
Ensure you use a computer to pre-format your SD card as FAT16. FAT16 was introduced by Microsoft a long time ago but it is simple and microcontrollers like simple. Many formatting programs support FAT16. Windows supports FAT16 natively with the format command.

It is suggested that you use the sdcard.org SD Memory Card Formatter app. It is available for PC and Mac.

<https://adafru.it/B2q>

<https://adafru.it/B2q>

Here is a shot of SD Card Formatter with a card inserted.



Reading SD Card Information

The program below is a slightly modified version of the Arduino example **SdFatInfo** program. The main change is to define the SPI select of the WIZ5500 Ethernet chip and set it high (unselected) so the sketch can talk to the SD card only.

For the Ethernet shield, put your formatted SD card into the SD card slot (top side up, don't force it, gentle). Put your shield on your Arduino-compatible board. Power the boards with a suitable power supply recommended for the Arduino. Connect the microcontroller board to your computer with a suitable USB cable.

For the Feathers, place the formatted micro-SD card into the micro-SD card slot in the Adalogger Feather. Plug the board into a USB port on your computer.

Using the Arduino IDE software and select the correct type of Arduino-compatible board and the serial port for the board in the Tools menu. You may have to push the reset button for the operating system to see the serial port. Load the following sketch, compile, and run.

The information about the SD card should be displayed on the serial monitor in the IDE.

```
/*
  SD card test for WIZ5500 Compatible Ethernet Boards

  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 not.

  The circuit:
  * SD card attached to SPI bus as follows:
  ** MOSI - pin 11 on Arduino Uno and Adafruit Metro
  ** MISO - pin 12 on Arduino Uno and Adafruit Metro
  ** CLK - pin 13 on Arduino Uno and Adafruit Metro
  ** CS - depends on your SD card shield or module (see below)

  created 28 Mar 2011 by Limor Fried
  modified 9 Apr 2012 by Tom Igoe
  modified 12 Apr 2018 by Mike Barela
  */
// 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;

// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit #2971 W5500 by Seeed Studio: Pin 4
// Sparkfun SD shield: pin 8
// Arduino Mega: Pin 53
// MKRZero SD: SDCARD_SS_PIN
const int chipSelect = 4;
//
// Chip Select for W5500 Ethernet (must be set as output in initialization)
const int W5500_SS = 10;
```

```

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.print("\nInitializing SD card...");

  // we'll use the initialization code from the utility libraries
  // since we're just testing if the card is working!
  pinMode(W5500_SS, OUTPUT); // set the Ethernet SS pin as an output (necessary!)
  digitalWrite(W5500_SS, HIGH); // but turn off the W5500 chip for now
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card 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) {
}

```

The serial monitor output below is for a micro-SD card formatted in Windows 10 for FAT. It is set for 60MB (it doesn't fill the card, it was an old Raspberry Pi Zero system card). It has several text files and a directory with a couple of files in it. Your Arduino Serial Monitor should display something similar.

The screenshot shows a serial monitor window titled "COM13 (Adafruit Feather 32u4)". The output text is as follows:

```

Initializing SD card...Wiring is correct and a card is present.

Card type: SDHC

Volume type is FAT16

Volume size (bytes): 62865408
Volume size (Kbytes): 61392
Volume size (Mbytes): 59

Files found on the card (name, date and size in bytes):
SYSTEM~1/      2018-04-20 09:24:54
  WPSETT~1.DAT  2018-04-20 09:24:54 12
  INDEXE~1     2018-04-20 09:24:58 76
SETUP.TXT      2007-03-22 05:40:06 437
LN2140.TXT     2018-04-20 09:28:34 856
LICENSE.TXT    2007-05-24 04:00:00 7575

```

At the bottom of the window, there are controls: a checked "Autoscroll" checkbox, a "Newline" dropdown menu, a "9600 baud" dropdown menu, and a "Clear output" button.

Listing Files

Once you are reliably reading SD card information, you can move on to looking at what files might be on the SD card. The code below is the Arduino SD Card Example Listfiles. It is modified to put the WIZ5500 chip select high so the SPI bus is only talking to the SD card.

At this point, please place a few files on the SD card from your computer to get an interesting output rather than using a blank card. It is suggested to put two or three text files with the filenames being no more than 8 characters before the period then end in .txt. An example would be README.txt or todo.txt. You can put the code text files on the card also, having them end in .ino will be fine also.

```
/* SDlistFiles

This example shows how print out the files in a directory on a SD card

The circuit:
* SD card attached to SPI bus as follows:
** MOSI - pin 11
** MISO - pin 12
** CLK - pin 13
** CS - pin 4 Adafruit #2971 and Metro/Uno

created   Nov 2010 by David A. Mellis
modified  9 Apr 2012 by Tom Igoe
modified  2 Feb 2014 by Scott Fitzgerald
modified 12 Apr 2018 by Mike Barela

This example code is in the public domain.

*/
#include <SPI.h>
#include <SD.h>

File root;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  pinMode(10, OUTPUT);    // set the SS pin as an output (necessary!)
  digitalWrite(10, HIGH); // but turn off the W5100 chip

  Serial.print("Initializing SD card...");

  if (!SD.begin(4)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

  root = SD.open("/");

  printDirectory(root, 0);

  Serial.println("done!");
}
```

```

    Serial.println("done!");
}

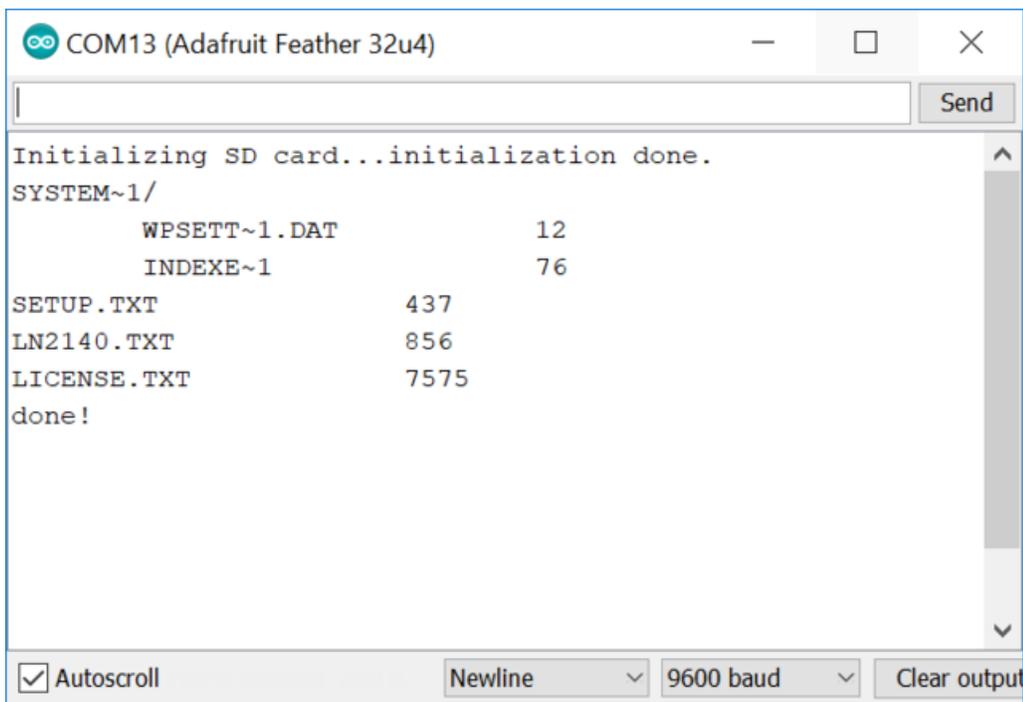
void loop() {
  // nothing happens after setup finishes.
}

void printDirectory(File dir, int numTabs) {
  while (true) {

    File entry = dir.openNextFile();
    if (! entry) {
      // no more files
      break;
    }
    for (uint8_t i = 0; i < numTabs; i++) {
      Serial.print('\t');
    }
    Serial.print(entry.name());
    if (entry.isDirectory()) {
      Serial.println("/");
      printDirectory(entry, numTabs + 1);
    } else {
      // files have sizes, directories do not
      Serial.print("\t\t");
      Serial.println(entry.size(), DEC);
    }
    entry.close();
  }
}
}

```

The screen shot below shows information on the same card as in the last sketch but lists the files and their sizes.



Serving Files over Ethernet

Now that all the SD card functionality appears to be working, the program below allows you to view the files on an SD card over the Ethernet interface and download any file you wish.

The program will provide a directory listing of the SD card to a web page. If a directory is clicked, the web page will show the contents of the subdirectory. If a file is clicked, the file will be sent to your computer. If you want to go back up the directory tree, use the browser back button.

The library which provides the WIZ5500 Ethernet functionality is called Ethernet2. The Adafruit version of Ethernet2 has been maintained and should be used instead of the Ethernet2 library available in the Arduino Library Manager. You can get the library code on GitHub at <https://github.com/adafruit/Ethernet2> (<https://adafru.it/plf>).

It is suggested that you put the Ethernet2 code from the Adafruit GitHub repository into your Arduino sketch folder, in the libraries subdirectory, in a subdirectory called Ethernet2.

For the source code below, you should review the following in the code:

- Byte array mac[] is set to a generic MAC address - if you have another device on your home network with the same address, change things a bit. Each value is an 8 bit value in hexadecimal (from 00 to FF).
- Byte array ip[] is the Internet Protocol (IP) address of an unused device on your home network. Having the Ethernet board way up at 177 (decimal) is fairly safe - if you know networking and this will not work, if you have your network on a network other than 192.168.1.xxx (say at 192.168.0.xxx or 10.0.0.xxx) change those also. You can find information like this from your Internet router.
- If you want to use DHCP to get an address from your router, uncomment the version of the Ethernet.begin call with only the mac address. It may be more difficult to get the address and the address might change if the router gives out a new address. So the default is a fixed address.

```
/*
 * SDWebBrowse.ino
 *
 * This sketch uses the microSD card slot on the a WIZ5500 Ethernet shield
 * to serve up files over a very minimal browsing interface
 *
 * Some code is from Bill Greiman's SdFatLib examples, some is from the
 * Arduino Ethernet WebServer example and the rest is from Limor Fried
 * (Adafruit) so its probably under GPL
 *
 * Tutorial is at https://learn.adafruit.com/arduino-ethernet-sd-card/serving-files-over-ethernet
 *
 * Please use the Adafruit Ethernet2 library https://github.com/adafruit/Ethernet2
 */

#include <SPI.h>
#include <SD.h>
#include <Ethernet2.h> // Using WIZ5500 Ethernet chip so Adafruit Ethernet2 library is required

/***** ETHERNET STUFF *****/
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // change if necessary
byte ip[] = { 192, 168, 1, 177 }; // change if necessary
EthernetServer server(80);

/***** SDCARD STUFF *****/
#define SDCARD_CS 4
File root;
```

```

#if defined(ESP8266)
  // default for ESPressif
  #define WIZ_CS 15
#elif defined(ESP32)
  #define WIZ_CS 33
#elif defined(ARDUINO_STM32_FEATHER)
  // default for WICED
  #define WIZ_CS PB4
#elif defined(TEENSYDUINO)
  #define WIZ_CS 10
#elif defined(ARDUINO_FEATHER52)
  #define WIZ_CS 11
#else // default for 328p, 32u4 and m0
  #define WIZ_CS 10
#endif

// store error strings in flash to save RAM
#define error(s) error_P(PSTR(s))

void error_P(const char* str) {
  Serial.print(F("error: "));
  Serial.println(str);

  while(1);
}

void setup() {
  Serial.begin(115200);
  while (!Serial); // For 32u4 based microcontrollers like 32u4 Adalogger Feather

  //Serial.print(F("Free RAM: ")); Serial.println(FreeRam());

  if (!SD.begin(SDCARD_CS)) {
    error("card.init failed!");
  }

  root = SD.open("/");
  printDirectory(root, 0);

  // Recursive list of all directories
  Serial.println(F("Files found in all dirs:"));
  printDirectory(root, 0);

  Serial.println();
  Serial.println(F("Done"));

  // Debugging complete, we start the server!
  Serial.println(F("Initializing WizNet"));
  Ethernet.init(WIZ_CS);
  // give the ethernet module time to boot up
  delay(1000);
  // start the Ethernet connection
  // Use the fixed IP specified. If you want to use DHCP first
  // then switch the Ethernet.begin statements
  Ethernet.begin(mac, ip);
  // try to configure using DHCP address instead of IP:
  // Ethernet.begin(mac);

  // print the Ethernet board/shield's IP address to Serial monitor

```

```

Serial.print(F("My IP address: "));
Serial.println(Ethernet.localIP());

server.begin();
}

void ListFiles(EthernetClient client, uint8_t flags, File dir) {
  client.println("<ul>");
  while (true) {
    File entry = dir.openNextFile();

    // done if past last used entry
    if (! entry) {
      // no more files
      break;
    }

    // print any indent spaces
    client.print("<li><a href=\"");
    client.print(entry.name());
    if (entry.isDirectory()) {
      client.println("/");
    }
    client.print(">");

    // print file name with possible blank fill
    client.print(entry.name());
    if (entry.isDirectory()) {
      client.println("/");
    }

    client.print("</a>");
    /*
    // print modify date/time if requested
    if (flags & LS_DATE) {
      dir.printFatDate(p.lastWriteDate);
      client.print(' ');
      dir.printFatTime(p.lastWriteTime);
    }
    // print size if requested
    if (!DIR_IS_SUBDIR(&p) && (flags & LS_SIZE)) {
      client.print(' ');
      client.print(p.fileSize);
    }
    */
    client.println("</li>");
    entry.close();
  }
  client.println("</ul>");
}

// How big our line buffer should be. 100 is plenty!
#define BUFSIZ 100

void loop()
{
  char clientline[BUFSIZ];
  char name[17];
  int index = 0;

```

```

EthernetClient client = server.available();
if (client) {
  // an http request ends with a blank line
  boolean current_line_is_blank = true;

  // reset the input buffer
  index = 0;

  while (client.connected()) {
    if (client.available()) {
      char c = client.read();

      // If it isn't a new line, add the character to the buffer
      if (c != '\n' && c != '\r') {
        clientline[index] = c;
        index++;
        // are we too big for the buffer? start tossing out data
        if (index >= BUFSIZ)
          index = BUFSIZ - 1;

        // continue to read more data!
        continue;
      }

      // got a \n or \r new line, which means the string is done
      clientline[index] = 0;

      // Print it out for debugging
      Serial.println(clientline);

      // Look for substring such as a request to get the file
      if (strstr(clientline, "GET /") != 0) {
        // this time no space after the /, so a sub-file!
        char *filename;

        filename = clientline + 5; // look after the "GET /" (5 chars) *****
        // a little trick, look for the " HTTP/1.1" string and
        // turn the first character of the substring into a 0 to clear it out.
        (strstr(clientline, " HTTP"))[0] = 0;

        if(filename[strlen(filename)-1] == '/') { // Trim a directory filename
          filename[strlen(filename)-1] = 0; // as Open throws error with trailing /
        }

        Serial.print(F("Web request for: ")); Serial.println(filename); // print the file we want

        File file = SD.open(filename, O_READ);
        if ( file == 0 ) { // Opening the file with return code of 0 is an error in SDFile.open
          client.println("HTTP/1.1 404 Not Found");
          client.println("Content-Type: text/html");
          client.println();
          client.println("<h2>File Not Found!</h2>");
          client.println("<br><h3>Couldn't open the File!</h3>");
          break;
        }

        Serial.println("File Opened!");

        client.println("HTTP/1.1 200 OK");

```

```

    if (file.isDirectory()) {
        Serial.println("is a directory");
        //file.close();
        client.println("Content-Type: text/html");
        client.println();
        client.print("<h2>Files in /");
        client.print(filename);
        client.println("</h2>");
        ListFiles(client,LS_SIZE,file);
        file.close();
    } else { // Any non-directory clicked, server will send file to client for download
        client.println("Content-Type: application/octet-stream");
        client.println();

        char file_buffer[16];
        int avail;
        while (avail = file.available()) {
            int to_read = min(avail, 16);
            if (to_read != file.read(file_buffer, to_read)) {
                break;
            }
            // uncomment the serial to debug (slow!)
            //Serial.write((char)c);
            client.write(file_buffer, to_read);
        }
        file.close();
    }
} else {
    // everything else is a 404
    client.println("HTTP/1.1 404 Not Found");
    client.println("Content-Type: text/html");
    client.println();
    client.println("<h2>File Not Found!</h2>");
}
break;
}
}
// give the web browser time to receive the data
delay(1);
client.stop();
}
}

void printDirectory(File dir, int numTabs) {
    while(true) {
        File entry = dir.openNextFile();
        if (! entry) {
            // no more files
            break;
        }
        for (uint8_t i=0; i<numTabs; i++) {
            Serial.print('\t');
        }
        Serial.print(entry.name());
        if (entry.isDirectory()) {
            Serial.println("/");
            printDirectory(entry, numTabs+1);
        } else {
            // files have sizes, directories do not

```

```
    // files have sizes, directories do not
    Serial.print("\t\t");
    Serial.println(entry.size(), DEC);
  }
  entry.close();
}
}
```

How it Works and Wrap-up

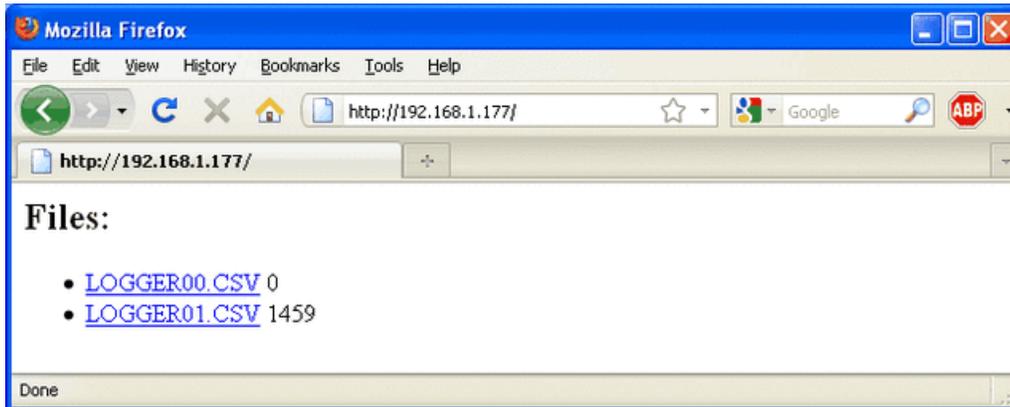
The main interface for the Ethernet uses code from the standard Arduino example WebServer and the SD card code used earlier in the tutorial.

The program implements the very bare bones of an HTML server. The code lists directories then files as an unordered HTML list.

Ensure the project is powered up and connected via an Ethernet cable to your home network. The yellow network activity light should blink now & then on one side of the connection, green on the other. If the lights are not on, check the Ethernet connection.

Open a web browser on a computer on your network and go to address <http://192.168.1.177> or to the alternate address you coded in your sketch. If you changed the sketch to use DHCP to get a free address from your router, the address received will be print on the serial monitor.

The program should output the heading **Files:** then a list of files on the card. You did put a few test files on the card earlier, yes? If not, no worries, power down the project, eject the SD card, put some text-based files on it, reinsert, and power the project up again.

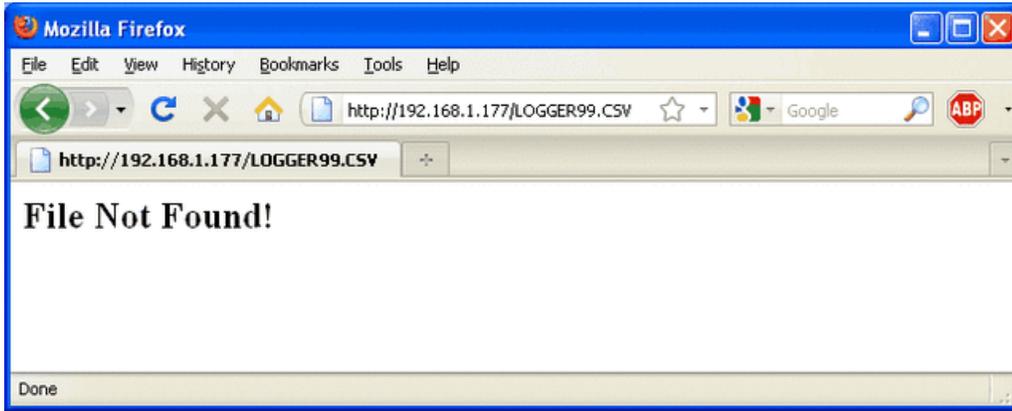


If you click on a file name, the sketch will send the file and the browser will ask you for a location to save the file.

If you click on a directory name, a new web page will appear displaying the files in the subdirectory.

Use the browser back button to go from the text listing screen back to the file list screen.

If there is a request for a file that is not on the SD card, the sketch will return an HTTP 404 error (file not found). Any other error will also return a 404 error.



Wrap-Up

When the original tutorial for using the Ethernet shield and sketch was written 7 years ago, serving files via Ethernet was pretty novel (read: wicked). Ethernet shields are not as common now. It is likely due to the proliferation of Wi-Fi boards.

Hopefully this updated tutorial will give you the basics for using a microcontroller to access an SD card (or something else) via Ethernet. The same methods can be used if you wire up a SD card breakout to the your microcontroller pins, just be sure to check for any necessary code changes.

Most newer microcontroller boards have more memory, so a more robust web server capability may considered. Perhaps you can look at this code and consider a new implementation with an Adafruit Express board and CircuitPython. The future awaits our innovation.

If you come up with great designs, please share them in the [Adafruit forums \(https://adafru.it/dYq\)](https://adafru.it/dYq) and in the [Adafruit Discord chat \(\)](#).