Using MicroSD Cards for Storage

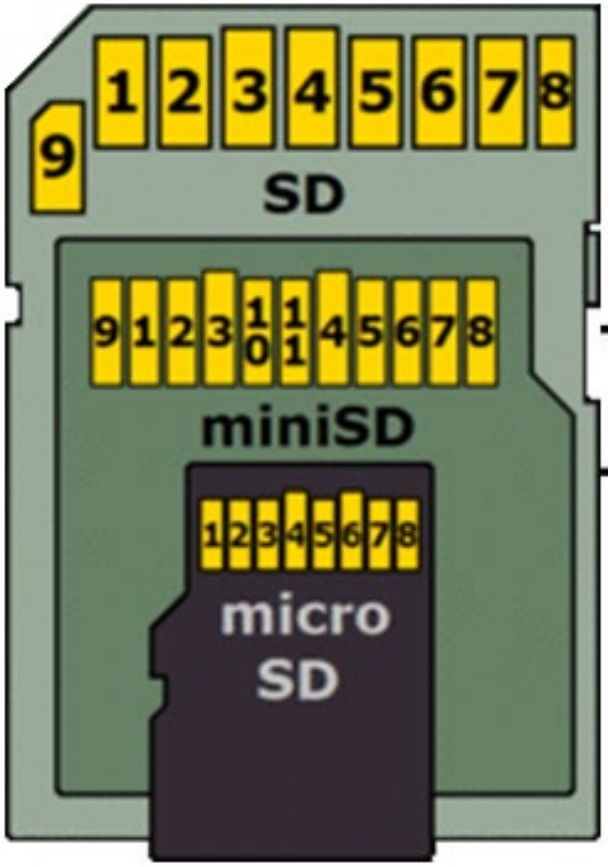
Using the SD Card, SD Library, and basics of data formats

Volatile vs. Nonvolatile

- There are two main types of memory:
  - Volatile and Non-Volatile
- Volatile Memory resets when power is lost
  - Ex. RAM on the Arduino where program variables are stored
- By contrast, Non-Volatile Memory keeps its data until overwritten or erased
  - Ex. Flash memory – where the code is uploaded on the Arduino (256 KB of storage)

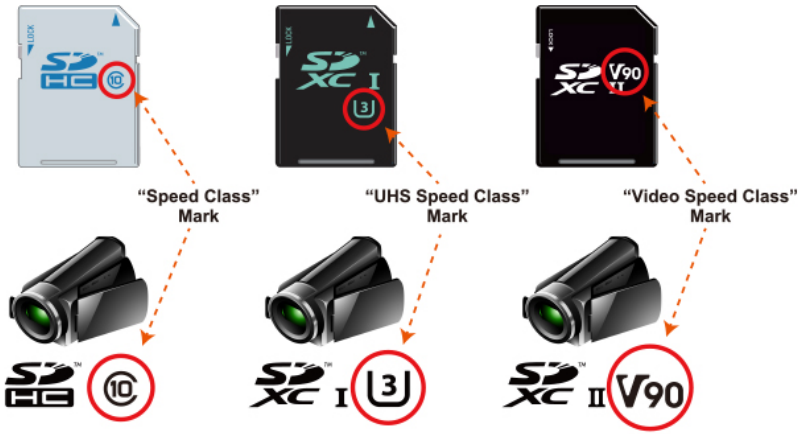
SD Cards

- The Secure Digital (SD) standard dates back to 1999
- In the early 2000s, SD storage was expensive for not much storage size
  - ~$365 for 64 MB of storage
- SD Cards are now inexpensive, lightweight, and small, so they are excellent for our needs
  - A 32 GB microSD card <$10
- Comes in 3 sizes
  - Full-sized – Often, the slot size on computers and older devices
  - Mini – Very uncommon
  - Micro – Most commonly used today



Class Markings

- SD Cards will have two types of markings, for speed and Storage Capacity
- Speed Markings (shown on the top)
  - These are not critical for our applications since the lowest speed is much faster than the Arduino
- The capacity will be marked by nothing (2 GB Max), HC(32 GB Max), XC(2 TB Max), or UC(128 TB Max)
  - These are important because they determine the **File System** and the Arduino Library is only compatible with two file systems

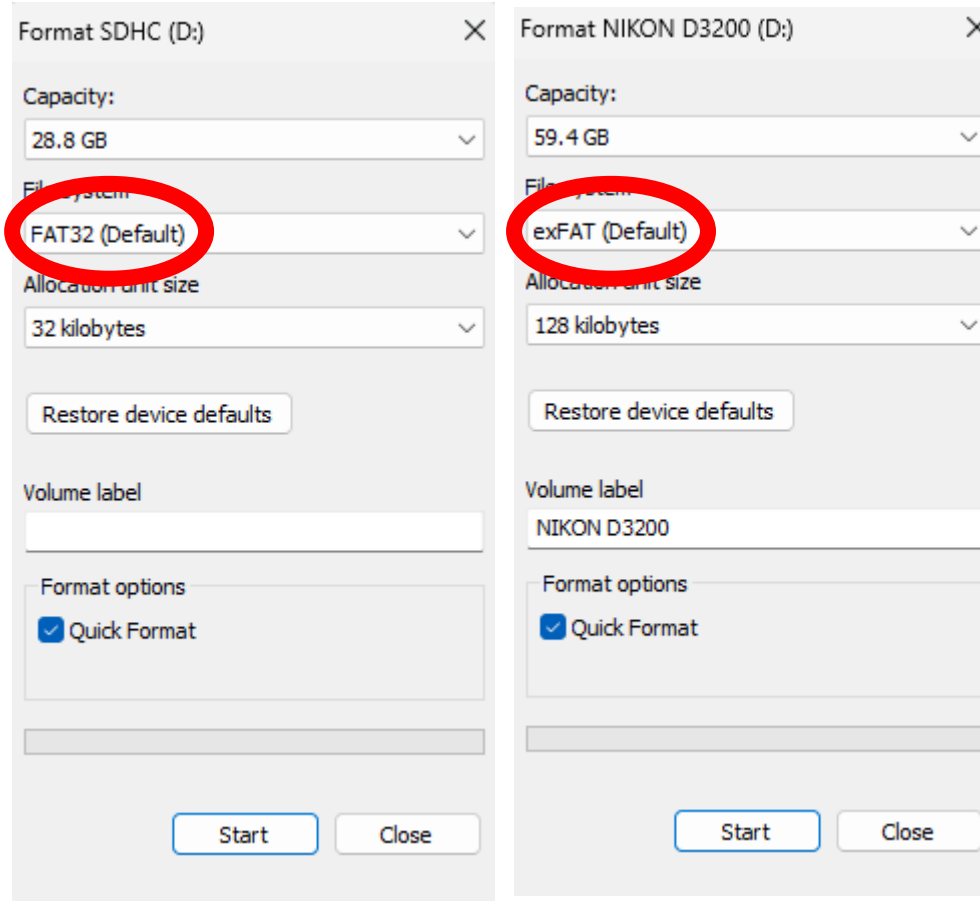


Speed markings


Capacity standard markings in increasing size

SD Cards File Systems

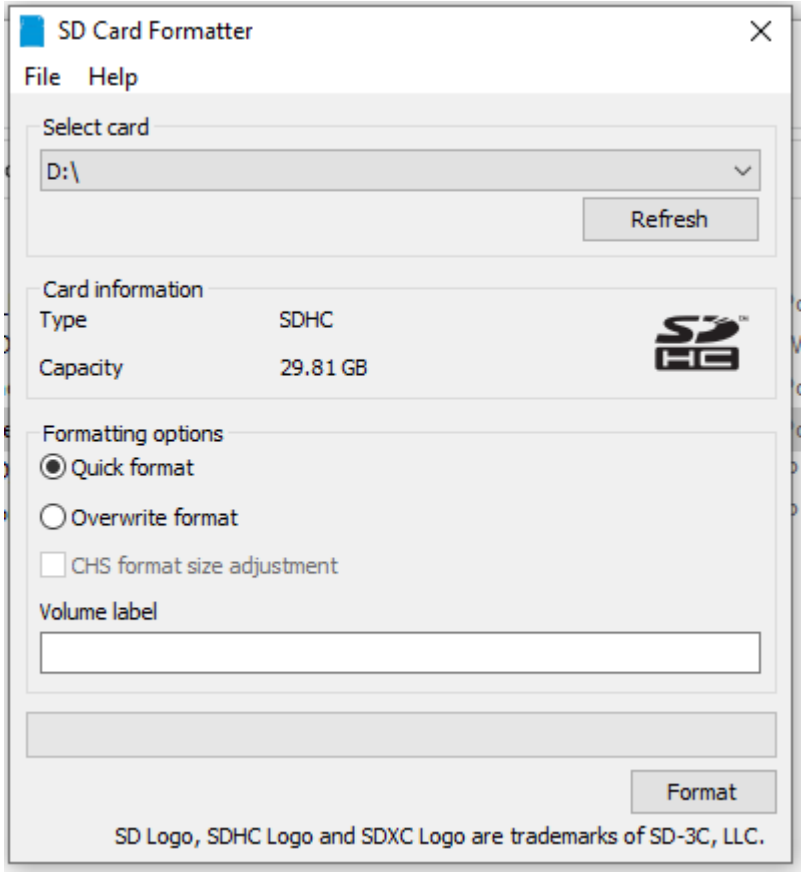
- An SD card (or any storage device) stores data in files and folders just like your computer
- The file system is how the data for those files is organized, accessed, written, etc.
- The file system places maximum limits on the total storage size and on the single file size
- For SD cards you are likely to encounter one of the versions of the **F**ile **A**llocation **T**able (FAT)
  - Total Size Under 2GB: FAT16
  - 2- 32 GB: FAT32
  - >32 GB: exFAT
- Some utilities will allow you to choose other formats, but they may not work properly with all cards



Two SD cards as shown in the Windows file format utility, notice the two different file systems

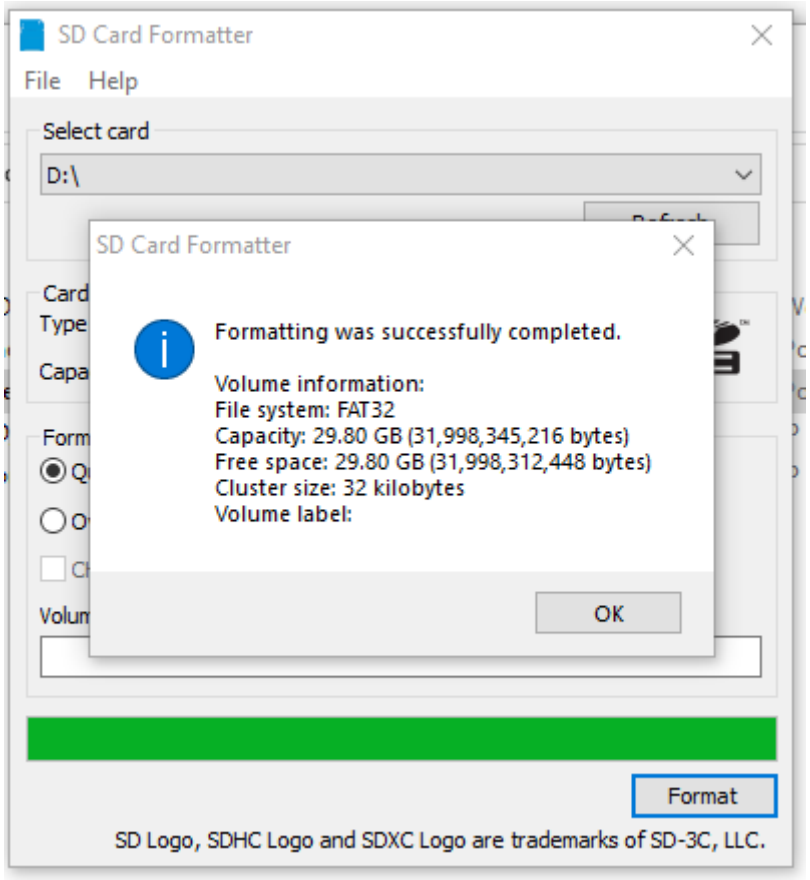
Using the SD Formatting Utility

- At some point, you may need to format the SD to reset it
  - A Downloadable Utility from the SD card Association
  - You operating system will also have some kind of formatting tool
- Formatting the card will completely erase all the data on the card
- When the SD card is inserted into the computer, the Card should appear under the **Select card** dropdown
  - Card Information will update, showing the size, label, and Capacity type
  - The label is just the text name the card displays and is not critical



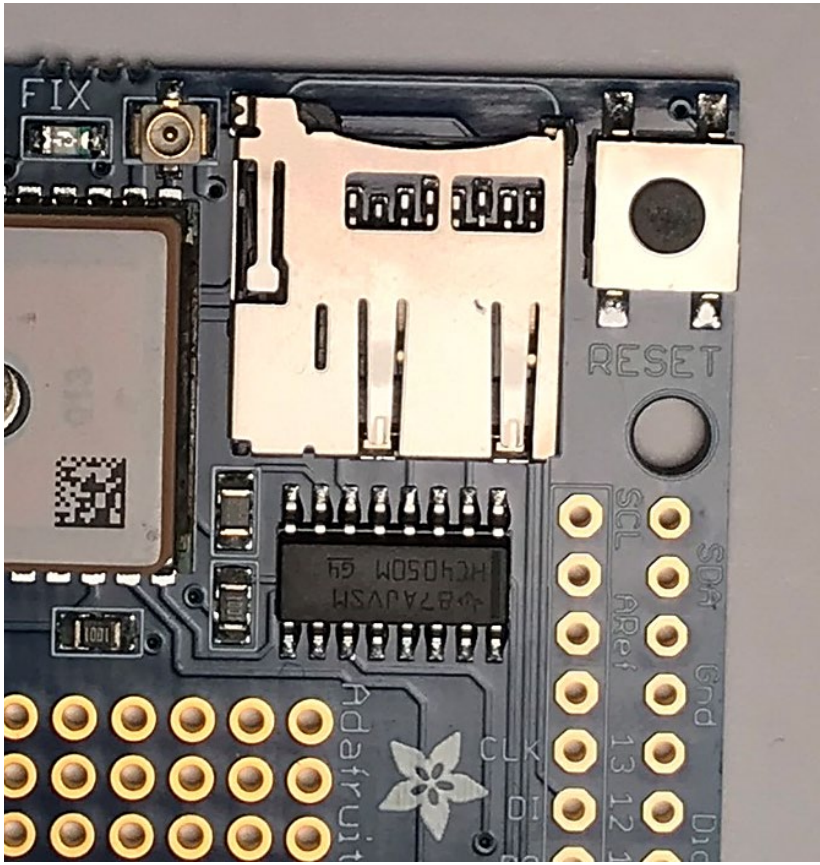
After Formatting

- The utility will automatically select the appropriate file system for the size card
  - It does not allow you to manually choose
- Quick formatting will just mark the old data as free to overwrite
- Overwrite will actually write blank data (takes longer but may be desired for security reasons)
- After formatting, you should get a completion message
  - Could fail, for example if the card is removed part of the way throug
- You can see here that the 32GB SD card was formatted in FAT32



SD Card Slot

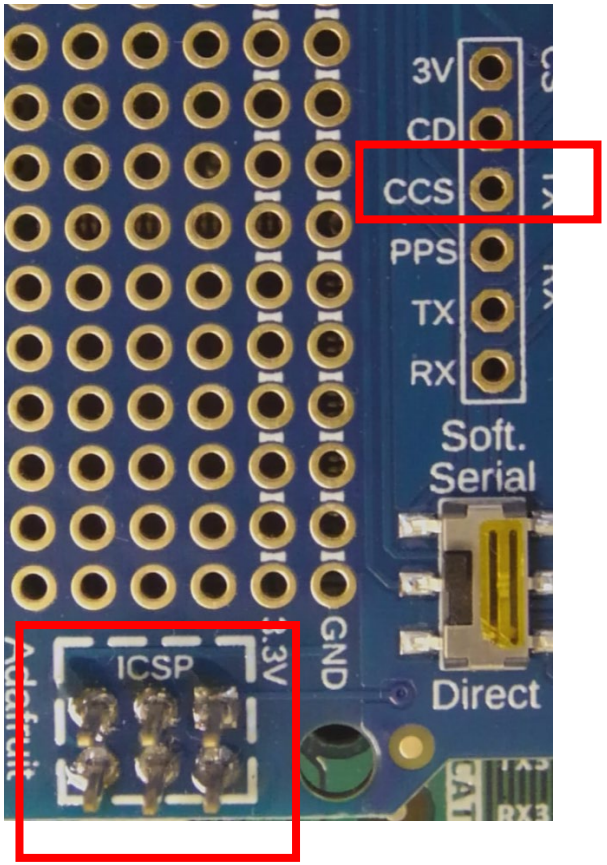
- The GPS shield has a microSD card slot
  - The card should be inserted contacts down, labeled side up
  - Cards are keyed and should fully slide in the correct orientation
- Communication between the Mega and the SD card uses SPI
  - The SD Card uses 3.3V Logic, the chip next to the slot is a level shifter
- When inserting the card, ensure that it latches with a click
  - To release, you just press the card in again
- If not fully inserted, the Arduino will not be able to communicate the card, no data will be logged
- To read the data, just eject the card and insert it into a computer (may need an adapter) and open the files



The Adafruit Ultimate GPS Logger Shield's microSD socket.

SD Card Physical Pins

- SPI communication requires 4 lines: MISO, MOSI, Clock, and Chip Select
- MISO, MOSI, and Clock are on the middle 6 pins connected to 50, 51, and 52
- CS is pin 10, on the right header
  - Although this is a free parameter in the library, only pin 10 is wired to the SD on the shield, using another pin would require wiring a jumper



SD Library: Basics

- The Library should have been installed with the Adafruit_GPS library and is just called SD
  - https://docs.arduino.cc/libraries/sd/
- The Library introduces two new variable types (classes)
- SD Class
  - This is used to set up communication and deal with the SD card and file system
- File Class
  - This is an individual file, so used for reading, writing, data, and closing
  - We can read data from the files using the library, but usually it is simpler just eject the card and look at the data on a computer

SD Library: Limitations

- Card must be FAT32 or FAT16
  - ExFAT cards will not be recognized
  - Maximum Storage size will be 32 GB (which should be sufficient for a full 4 hours of flight time)
- File names must follow the 8.3 Name Format
  - (Up to) 8 Characters – a period – 3 more characters
  - Letters, numbers, dashes (-), and underscores _ : Ok
  - Avoid spaces, additional periods, special characters ($, &, *, etc.)
  - Recommend numbering your data files (DATA0.csv, DATA1.csv, DATA2.csv …) or using times (GPS_1215.dat, GPS_1225.dat, ..)
  - The 3 characters after the period are called a file extension
- The SD library can work with subfolders, but it is not necessary or recommended

SD Library: SD Important Functions

- begin(CS_PIN)
  - Used to start communication with SD card, you must give it the correct pin for the Chip Select (which will be 10 for the Adafruit_GPS) – **Will not error if you forget the CS Pin, will instead just fail to communicate**
  - Returns either a 1 (success) or 0 (failure)
- open (Filename, mode)
  - Filename is the name you would like to open, can be an existing file or a new file (which will create the file)
  - Mode is used to open a file for reading or writing; two constants exist **FILE_WRITE** and **FILE_READ**
  - Will return a File variable for the opened file, so will want to place that in a variable to actually write data
- exists(Filename)
  - Determines if the file already exists on the SD card

SD Library: File Functions

- print(data) and println(data)
  - These functions work similarly to the serial port functions, but instead of writing the data to the file they are called on, write the data as character strings by default, just like Serial.print()
  - println() adds the <cr><nl> so that the next data will be written on a new line in the file
- flush()
  - By default, the library saves the characters to a buffer and waits until full to send to the SD card
  - This function forces any data to be sent to the SD card when called, similar to forcing a save
- close()
  - Flushes any data and then closes the file
  - Will have to call open again before writing new data

General Process Data to SD Card

- SD card communication initialized in setup()
  - SD.begin(CS);
- Open/create SD file
  - myFile = SD.open(filename, FILE_WRITE);
- Write data to the file – same as printing to Serial Monitor
  - myFile.println("This sentence will be written to my SD file");
  - May want to occasionally call flush()
- When finished, close the file
  - myFile.close()

File Extensions

- The file extension just tells your computer what program to use to try and open the file (.docx for Word, .xlsx for Excel, .mp4 for VLC media player)
- We will be saving our data as a string of characters rather than the raw binary bytes
  - We can always open the data in Notepad to look at it or copy it, but we can make our lives easier with the .csv extension
- CSV is a comma-separated values file
  - Will open in Excel by default, you can then save as XLSX to save formulas or graphs
  - Each line of text will be a separate row
  - Commas separate the data into separate columns

```
1  ***Logging in file 01153040.csv created at
2  START,Timestamp,Altitude,# Satellites,Fix Q
3  START,_____No Fix!_____,0.00,0,0,5142,0,0
4  START,_____No Fix!_____,0.00,0,0,10142,0,
5  START,_____No Fix!_____,0.00,0,0,15142,0,
6  START,_____No Fix!_____,0.00,0,0,20142,0,
7  START,09/01/2019 15:31:04,1204.10,6,1,28116
8  START,09/01/2019 15:31:09,1207.20,6,1,33116
9  START,09/01/2019 15:31:14,1206.80,6,1,38116
10 START,09/01/2019 15:31:19,1208.10,6,1,43116
11 START,09/01/2019 15:31:24,1209.80,6,1,48116
```

	A	B	C	D	E	F	G
1	***Logging in file 01153040.csv created at 09/01/2019 15:30:40 for code version FlightC						
2	START	Timestamp	Altitude	# Satellite	Fix Qualit	FC Millis	LC Upper
3	START	_____No Fix!_____	0	0	0	5142	0
4	START	_____No Fix!_____	0	0	0	10142	0
5	START	_____No Fix!_____	0	0	0	15142	0
6	START	_____No Fix!_____	0	0	0	20142	0
7	START	9/1/2019 15:31	1204.1	6	1	28116	0
8	START	9/1/2019 15:31	1207.2	6	1	33116	0
9	START	9/1/2019 15:31	1206.8	6	1	38116	0
10	START	9/1/2019 15:31	1208.1	6	1	43116	0
11	START	9/1/2019 15:31	1209.8	6	1	48116	0

This is an example of a .csv file. Left – opened in a notepad. Right – opened in Excel, the commas are used to separate the columns.

Designing a Data Format

- You will choose how you organize and write your data in your files
- Plain text allows you to easily view the data and can often spot if an error occurred during writing
- At the beginning of each file, write the column names as the first line
- Consider a start and stop marker (could literally be “START” and “STOP”) so you can tell if data finished writing
- Think about decimal places, by default print() will show 2 for floats, do you need + and – signs
- If your data points become disordered (a software error caused you to write data on the first file), is there anything in the data to allow you to resort (timestamp each datapoint)?
  - Maybe a column for filenumber, a column for data point, etc.
- How much space does your data take up, is your SD card large enough

```
1  Num1,Num2,Num3,Num4
2  1.50,3.25,5.00,7.57,
3  2.00,3.75,5.50,8.07,
4  2.50,4.25,6.00,8.57,
5  3.00,4.75,6.50,9.07,
6  3.50,5.25,7.00,9.57,
7  4.00,5.75,7.50,10.07,
8  4.50,6.25,8.00,10.57,
```