Summary:

This activity will walk students through how to save data onto an SD card using the Adafruit Ultimate GPS Logger Shield. By the end of this activity, students will understand how to save data to an SD card, read data from an SD card, and create an csv file.

Materials:

Each student should have the following materials, equipment and supplies:

- Computer with Arduino IDE installed and microSD or SD reader
- USB-AB programming cable
- Arduino Mega microcontroller with assembled Adafruit Ultimate GPS Logger Shield attached
- microSD card
- microSD to SD adapter (if computer has an SD reader instead of a microSD reader)

Procedure:

Activity A: Download Adafruit SD Library

1. In order to save data to the SD card using the Arduino Mega, the SD library will be required. It can be found at https://github.com/adafruit/SD. Download it.
    a. It is currently version 3.

2. This folder will download as a zip folder. Extract it.

3. Open the extracted folder. Inside, there will be another folder named "SD-master." Rename it as "SD".

4. Move the "SD" folder to the libraries folder for the Arduino IDE.
    a. This should be found at Documents >> Arduino >> libraries

5. Restart the Arduino IDE if open.

Activity B: Initialize Communication with SD Card

1. Start a new sketch. Save this sketch as *SDcardWriteIntro.io*.

2. To use SPI to communicate with the SD card, two libraries need to be included. Figure 1 shows these libraries and how to include them. These lines of code should be written before setup().

```
/* Include statements for required libraries ***********************************************/
#include <SPI.h>      // This is the SPI library for communication with the SD card
#include <SD.h>       // SD library for writing/reading files on the SD card
```

Figure 1: Shown are the include statements for the SPI library and the SD library. The SPI library is used to communication via SPI with the SD card. The SD library is used to read/write data on the SD card.

3. The multiple communication lines for SPI need to be defined. Chip select, MOSI, MISO, and SCLK are pins 10, 11, 12, and 13, respectively. These will be used when the SPI communication initiated. Figure 2 shows how to define these variables.

```
/* SD Card & Files *************************************************************************/
#define CS 10         // SPI chip select pin
#define MOSI 11       // SPI master out, slave in pin
#define MISO 12       // SPI master in, slave out pin
#define CLK 13        // SPI clock pin
```

Figure 2: Shown is how to define the numbers that will be used for the SPI pins.

4. In setup(), begin Serial communication with the Serial Monitor.

5. Just like serial communications, SPI communication with the SD card is started in setup(), as shown in Figure 3.

```
  /* Start communications with peripherals ***************************************************/
  Serial.begin(9600);                           // Communicate with Serial Monitor

  Serial.println("Initializing SD card...");
  if (!SD.begin(CS, MOSI, MISO, CLK)) {         // Initialize communication with SD card
    Serial.println("Initialization failed :("); // If initialization failed, print it
    while(1);                                   //    and enter a forever while loop
  }
  Serial.println("Initialization successful!"); // If initialization successful, print it
}
```

Figure 3: Shown is how to initialize communication with the SD card in setup(). If successful, "Initialization successful :)" will be printed to the Serial Monitor.

6. Insert the microSD card into the slot on the Shield. There should be a click when it latches.

7. Plug the Mega into the computer and upload the sketch. In the Serial Monitor, one of panels in Figure 4 will be shown. If the Serial Monitor shows that initialization was successful, proceed to Activity C. If it shows that the initialization failed, proceed to step 8.

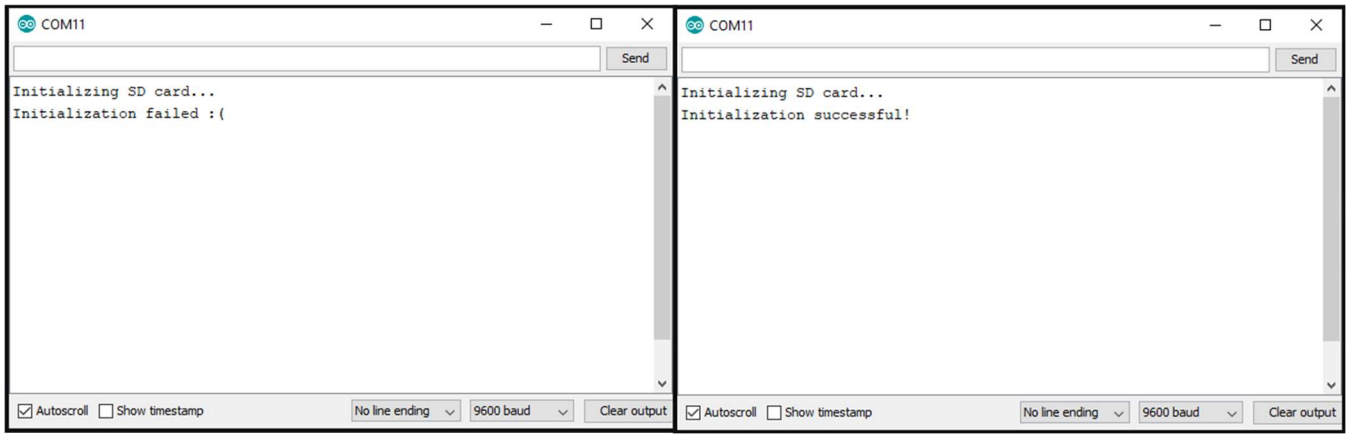


Figure 4: Left – the Serial Monitor if the SD card failed to initialize. Right – the Serial Monitor if the SD card initialization was successful.

8. There are a variety of reasons why the SD card communication did not initialize. Some common reasons are: SD card is not fully inserted (make sure the click is heard), libraries not installed, wrong chip select defined, or SPI pins not wired properly. Try these one by one and then reupload the sketch. If the issue persists, ask for the help of an instructor.

Activity C: Write to SD Card

6. Figure 5 shows how to define a file object. Whenever something is to be done with an SD file, such as opening, closing, creating, deleting, writing to, or reading from this file object will be used. A char array variable needs to also be created for the file name.

```
/* SD Card & Files ******************************************************************************/
#define CS 10                  // SPI chip select pin
#define MOSI 11                // SPI master out, slave in pin
#define MISO 12                // SPI master in, slave out pin
#define CLK 13                 // SPI clock pin
File myFile;                   // File object
char fileName[21] = "test.txt"; // Name of file to be created on SD card
```

Figure 5: Shown is how to define a file object and a char array filename. These will be used when communicating with the SD card.

7. For now, everything will be done in setup(). To create a file, the file object defined above will be used. Figure 6 shows how to create/open a file.
    a. The first input into SD.open() is the file name for the created file. The second input is what behavior the sketch will perform on this file.
    b. File names must follow a 8.3 format. This means the name can be up to 8 characters long and the extension is 3 characters. Ex: test1234.txt

```
/* Open, write to, and close an SD file ************************************************/
myFile = SD.open(fileName, FILE_WRITE);
```

Figure 6: This is the command that will open an SD file.

8. Writing to the SD card is as easy as printing to the Serial Monitor. Instead of Serial.println(), the command is myFile.println(). Using this, write three separate sentences to the SD file: "Hello World!," "This is a test," and "Science is cool."

9. Between each of the sentences, flush the data. This command is myFile.flush().
    a. This command makes the Arduino waits for the transmission of outgoing data to complete before proceeding.

10. The last thing that must be done is the file needs to be closed. This command is simple, it is myFile.close(). Insert this command at the end of setup() followed by "Done!" being printed to the Serial Monitor. Upload the sketch.

11. Once "Done!" has shown up on the Serial Monitor remove the microSD card from the Arduino.

12. Insert the SD card into the computer, using the microSD to SD adapter if necessary, and open the file created on the SD card.

13. Confirm the contents of the file created are as expected. Safety remove the SD card from the computer.

14. Edit the sketch to use a different filename and to use myFile.print() instead of myFile.println(). Reupload the sketch.

15. After the sketch has completed, remove the SD card and open the new file on the computer. Does this change make sense?

Activity D: Create CSV

1. Open *SDcardWriteInfo.io*. Change the filename of the created file to "csvtest.csv" and create an array (with 4 entries) of floats. Initialize these values to {1.5, 3.25, 5.0, 7.57}.

2. Edit setup() so that it begins communications with the Serial Monitor and the SD card.

3. In loop(), have the follow actions performed in order: open the SD card file to write; write these values, separated by commas, to the SD card (see Figure 7 for an example) on a new line; flush; close the SD file; add 0.5 to all values of the array; print "Done!" to the Serial Monitor; and delay for 3 seconds.

```
51  if (myFile) {
52    Serial.println("Open successful for " + String(fileName));
53
54    for (int i=0; i<4; i++) {
55      myFile.print(testArray[i]);
56      myFile.print(",");
57    }
58    myFile.println("");
59    myFile.flush();
60  }
61  else Serial.println("There was an error opening " + String(fileName));
```

Figure 7: Shown is how to write all the values from a length 4 array with commas separating them. The println call after the for loop starts a new line in the file.

4. Upload the sketch. Wait until "Done!" has printed to the Serial Monitor at least 5 times and then remove the SD card.

5. Open the CSV file in Excel as well. It should look like Figure 8. Having the CSV format makes looking at data and analyzing data in Excel much easier because Excel separates it for the user.

	A	B	C	D
1	1.5	3.25	5	7.57
2	2	3.75	5.5	8.07
3	2.5	4.25	6	8.57
4	3	4.75	6.5	9.07
5	1.5	3.25	5	7.57
6	2	3.75	5.5	8.07
7	2.5	4.25	6	8.57
8	3	4.75	6.5	9.07
9	3.5	5.25	7	9.57

Figure 8: This is what the csv file should look like when opened in Excel. Excel does the parsing automatically, so the data is easy to look at and plot.

Activity E: Combining GPS and SD Functionality

Note: NMEA Sentences are very nice for CSV files because their information is already comma separated. The purpose of this activity is to save NMEA sentences to an SD file.

1. If not already, jumper the Shield to the Mega to allow for Serial communication for the GPS module. Create a new sketch called *NMEA_SD.io*. Clear the SD card.

2. Using previous activities or sketches as a reference, set up communication with GPS and the SD card. Have the GPS send RMC sentences.

3. When a new NMEA sentence is received, write it to the SD card. Don't forget to flush after writing.

4. Upload the sketch. Look at the file created on the computer, it should look like Figure 9.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	$GPRMC	224936.067	V					0	0	61119			N*4A
2	$GPRMC	224937.067	V					0	0	61119			N*4B
3	$GPRMC	224938.067	V					0	0	61119			N*44
4	$GPRMC	224939.067	V					0	0	61119			N*45
5	$GPRMC	224940.067	V					0	0	61119			N*4B
6	$GPRMC	224941.067	V					0	0	61119			N*4A
7	$GPRMC	224942.067	V					0	0	61119			N*49
8	$GPRMC	224943.067	V					0	0	61119			N*48
9	$GPRMC	224944.067	V					0	0	61119			N*4F
10	$GPRMC	224945.067	V					0	0	61119			N*4E
11	$GPRMC	224946.067	V					0	0	61119			N*4D

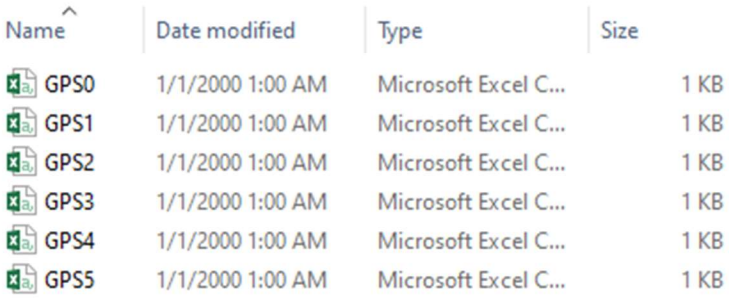
Figure 9: This is what the RMC csv file should look like when it's opened in Excel. In this case, the GPS didn't have a fix, so some columns are empty.

5. Create a integer variable called count. After a NMEA sentence is written to the SD card, increment this variable by 1 and print it to the Serial Monitor.

6. Once the variable has reached a value of 20, close the SD file and open a new one. Have "New file created: XXXX.csv" print to the Serial Monitor when a file is created.
    a. XXXX.csv is the filename. This filename needs to change every time a new file is created.

7. Upload the sketch and have it create at least 4 files before removing the SD card.

8. The files on the SD card should look something like the left panel of Figure 10. The information inside the file should look something like the right panel of Figure 10.



Name	Date modified	Type	Size
GPS0	1/1/2000 1:00 AM	Microsoft Excel C...	1 KB
GPS1	1/1/2000 1:00 AM	Microsoft Excel C...	1 KB
GPS2	1/1/2000 1:00 AM	Microsoft Excel C...	1 KB
GPS3	1/1/2000 1:00 AM	Microsoft Excel C...	1 KB
GPS4	1/1/2000 1:00 AM	Microsoft Excel C...	1 KB
GPS5	1/1/2000 1:00 AM	Microsoft Excel C...	1 KB

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	$GPRMC	230017.402	V					0	0	61119			N*42
2	$GPRMC	230018.402	V					0	0	61119			N*4D
3	$GPRMC	230019.402	V					0	0	61119			N*4C
4	$GPRMC	230020.402	V					0	0	61119			N*46
5	$GPRMC	230021.402	V					0	0	61119			N*47
6	$GPRMC	230022.402	V					0	0	61119			N*44
7	$GPRMC	230023.402	V					0	0	61119			N*45
8	$GPRMC	230024.402	V					0	0	61119			N*42
9	$GPRMC	230025.402	V					0	0	61119			N*43
10	$GPRMC	230026.402	V					0	0	61119			N*40
11	$GPRMC	230027.402	V					0	0	61119			N*41
12	$GPRMC	230028.402	V					0	0	61119			N*4E
13	$GPRMC	230029.402	V					0	0	61119			N*4F
14	$GPRMC	230030.402	V					0	0	61119			N*47
15	$GPRMC	230031.402	V					0	0	61119			N*46
16	$GPRMC	230032.402	V					0	0	61119			N*45
17	$GPRMC	230033.402	V					0	0	61119			N*44
18	$GPRMC	230034.402	V					0	0	61119			N*43
19	$GPRMC	230035.402	V					0	0	61119			N*42
20	$GPRMC	230036.402	V					0	0	61119			N*41

Figure 10: Left – This is what the files on the SD card should look like. Right – This is what an individual file on the SD card should look like when it's opened on Excel. Again, the GPS did not have a fix for this run, so some columns are empty.