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.
    a. It is currently version 3.

2. Use the green CODE button to download a zip file of the archive.

3. Open the zip file. Inside, there will be another folder named "SD-master." Copy the folder into the libraries folder for the Arduino IDE. Rename the folder to just "SD".
    a. Remember this is inside your Sketchbook folder set in
    b. This should be found at Documents >> Arduino >> libraries

4. Restart the Arduino IDE if open.

Activity B: Initialize Communication with SD Card

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

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.

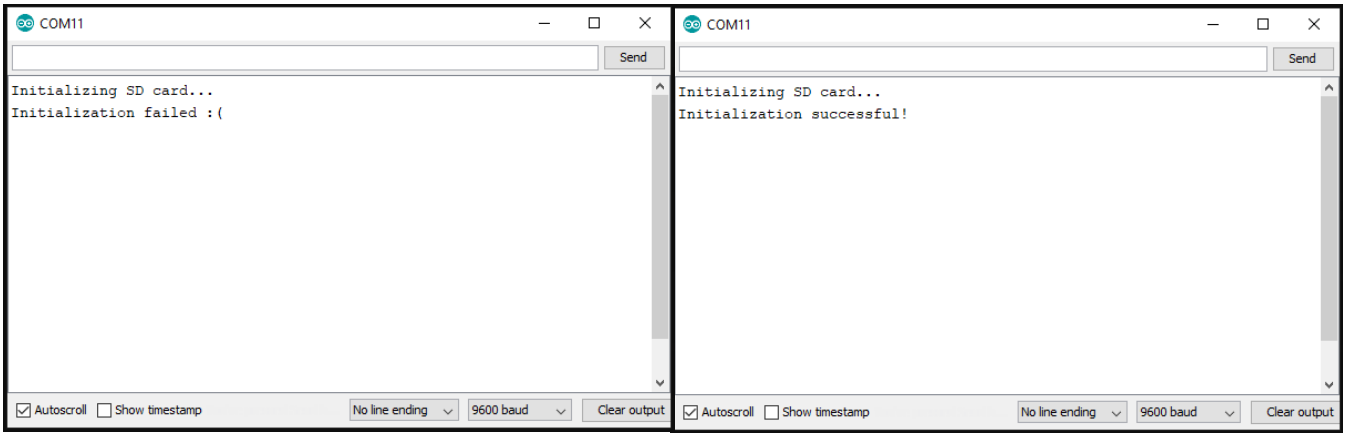

Figure 4: Left – the Serial Monitor if the SD card failed to initialize. Right – the Serial Monitor if the SD card initialization was successful.

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

5. 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.

6. 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

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

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

7. 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."

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

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

  if (myFile) {
    Serial.println("Open successful for " + String(fileName));
    myFile.println("Hello World!");
    myFile.flush();
    myFile.println("This is a test");
    myFile.flush();
    myFile.println("Science is cool.");
    myFile.flush();
  }
  else Serial.println("There was an error opening " + String(fileName));

  myFile.close();
  Serial.println("Done!");
}
```

Figure 7: Shown is how to write data to the files after the file has been opened. Note that the flush() is not strictly necessary and but it is good practice to ensure the write occurs at that time.

9. 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. Note in this case our loop does nothing

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

11. Insert the SD card into the computer, using the microSD to SD adapter if necessary, and open the file created on the SD card. You should see a single file called test.txt

12. Confirm the contents of the file created are as expected.

```
1  Hello World!
2  This is a test
3  Science is cool.
4
```

Figure 8: Shown here is the output that should be in the text file is generated from the above code.

13. Edit the sketch to use a different filename and to use myFile.print() instead of myFile.println(). Reupload the sketch. Try making your own output. Don't forget to eject the SD card from your computer when you remove it to put it back in the Arduino. Also experiment with writing integers or floats to the files.

14. After the sketch has completed, remove the SD card and open the new file on the computer. Does this change make sense? Remember your filenames need to follow the 8.3 rules.

15. Also notice what happens if you reset the Arduino without deleting the file or previous data.

```
1  Hello World!
2  Hello World!
3  This is a test
4  Science is cool.
5  Hello World!
6  This is a test
7  Science is cool.
8  Hello World!
9  This is a test
10 Science is cool.
11
```

Figure 9: The output to the SD card after several presses of the reset button. Notice that each time the data is appended the file and not overwritten. It is even possible that a reset may occur the write has completed. When writing your program you want consider how you data will resume on a power loss as your payload could easily have a momentary power loss during flight.