Brief Documentation for CRTLfinal.ino (final Arduino program version).

**Operation:**

The SD card must be plugged in to the SD shield *before* the Arduino Leonardo is powered. This is because the operation to initialize the SD card can only be called once—if it fails, the program cannot try starting the SD card again.

All “F0.txt” files on the SD card should be cleared before Arduino operation. If these files are not cleared beforehand, then the new files that are created will start from the first available number, like “F87.txt.” This is problematic when AnalyzeCRData.py will be run, because then the detections from all files will be combined somehow.

**Context:**

The program is divided into many functions to make the main code easier to understand. Some of these functions are for reading data from the GPS, and another function is for reading the HEX timestamp from the circuit board clock system. It is not a good idea to try reading data from the GPS all of the time, because GPS data only arrives once a second (and is not necessarily synchronized with the pulse-per-second (PPS) signal). The Arduino would have to keep waiting for the next string of GPS data to arrive. To understand the time constraints of the Arduino and its capabilities, let me introduce the concept of *blocking*. Blocking is when the Arduino is busy doing one thing and is unable to do another thing. Blocking is a problem for the Arduino because the Arduino can only execute one “thread” or program at a time. If you need the Arduino to do two things simultaneously, then you will need two Arduinos to do those separate tasks.

There are ways to circumvent or ignore blocking. For example, some Arduino functions like digitalRead() take only 4μs. The “save button” on the Arduino is a button that is pressed whenever the Arduino is about to be powered off—this button instructs the Arduino to save the data it is working with so that the user can collect the data without any data loss. But how does the Arduino check to see if the save button is pressed all of the time? Well, checking the save button has been added to a “queue” of things that the Arduino does while it is busy doing other things. If no detections are arriving, then the Arduino goes between checking for the trigger signal and checking for a save button signal. These things do not happen simultaneously. They happen sequentially, and these tasks are fast enough that either signal is never missed.

Time is always precious in this program because we are always trying to reduce dead time while preserving the accuracy and integrity of our data. Because of time constraints, it is superfluous to try (for example) converting the timestamps that are in HEX to their decimal form. We can always do that processing later.

**Program Outline:**

**Setup:**

* Arduino powers on.
* Arduino tries to initialize the SD card with a speed of 5MHz. The SD cards in our lab have a speed rating of 10MHz, this is the number that is circled on the SD card. I do not believe that SD card speed has a significant impact on dead time.
* Arduino turns pins A3-A5 into output pins and sets their initial voltages.

**Main Loop (this repeats from the beginning for every file):**

* Wait up to 1 second for the next GPS PPS signal to arrive (don’t record while there is no GPS signal).
* Generate an unused file name.
* Create the new file and open it — the file stays open for 5000 detections.
* Record a GPS string through D11. Record the circuit board’s clock time immediately after, even though there was no detection.

**Ordinary Operation (99% of the time):**

* + Now, if the trigger signal goes high, record the time from the circuit board. This timestamp represents a muon detection.
  + If the save button signal goes low, save the current file and stop recording.
  + Otherwise, start the Main Loop again once 5000 detections are reached.