Hardware Required

In this tutorial we'll learn how to write the data we gather out to an SD card that can be read on your PC when the rocket returns

-Micro SD Card Shield:

- Purchase: https://www.sparkfun.com/products/9802
- Schematic: https://www.sparkfun.com/datasheets/DevTools/Arduino/microSD_Shield-v13%20Schematic.pdf

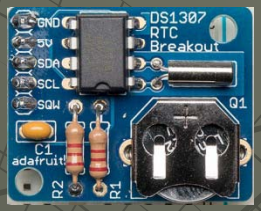
-SD Card Shield with Real Time Calendar Clock:

- Purchase: https://www.adafruit.com/products/1141
- Schematic: https://learn.adafruit.com/assets/9094

-Arduino Uno: From Arduino, Amazon, Sparkfun, MP3Cars, many more:

- Purchase: http://www.amazon.com/Arduino-UNO-board-DIP-ATmega328P/dp/B006H06TVG
- Schematic: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

Hardware Required

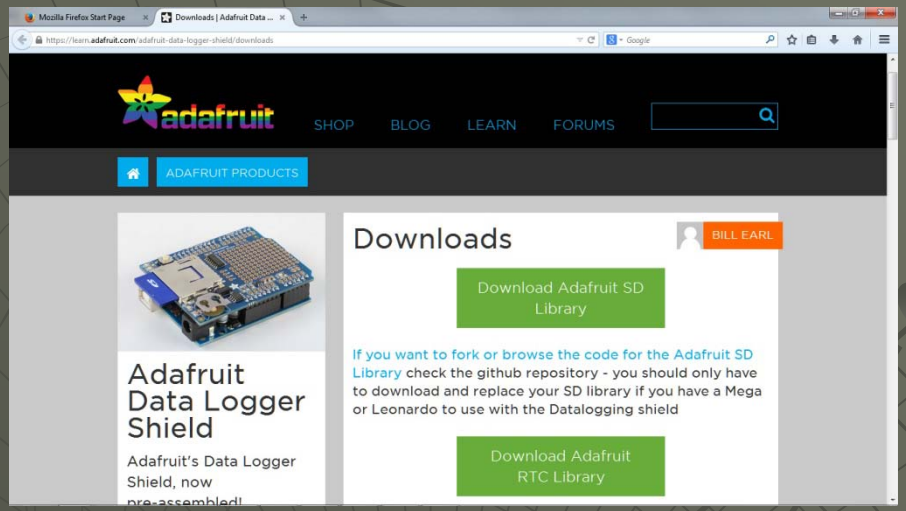


- Micro SD Card Breakout Board:
- Purchase: https://www.adafruit.com/products/254
- Schematic: https://github.com/adafruit/MicroSD-breakout-board/blob/master/usdbreakout.png

- Real Time Calendar Clock Breakout Board:
- Purchase: https://www.adafruit.com/products/264
- Schematic: https://github.com/adafruit/DS1307-breakout-board/blob/master/ds1307.png
- Assembly Instructions: https://learn.adafruit.com/downloads/pdf/ds1307-real-time-clock-breakout-board-kit.pdf

- Micro SD Card with adapter
- Purchase: http://www.amazon.com/SanDisk-Class-microSDHC-Memory-Adapter/dp/B001EHHVPA/ref=sr_1_10

- A PC or laptop



Software Required

Arduino Integrated Development Environment (IDE):
http://arduino.cc/en/main/software#.Uy4WgU1OUpA

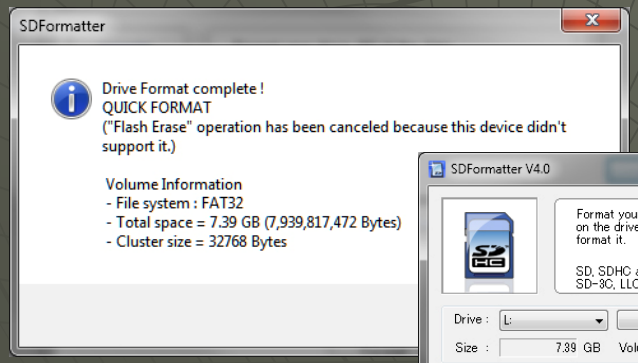
SD Card Formatter:
https://www.sdcard.org/downloads/formatter_4/

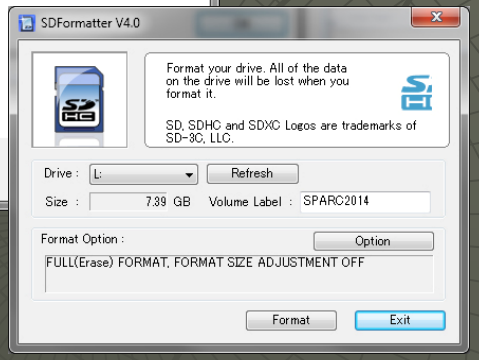
Real Time Calendar Clock Library from Adafruit (NOTE: we will use the SD Card Library in the IDE)
https://github.com/adafruit/RTClib/archive/master.zip

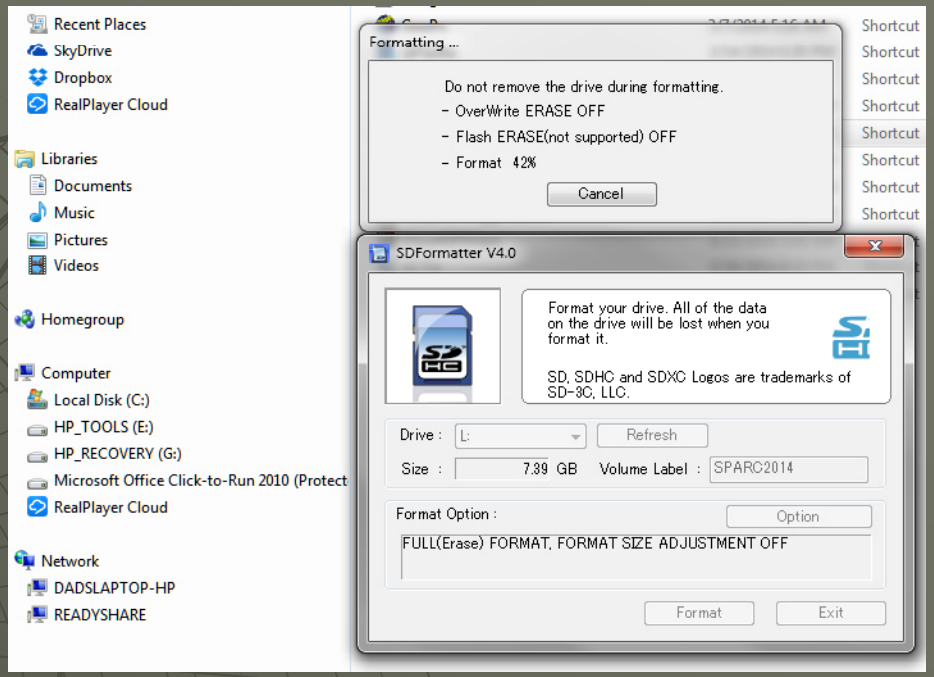
Sketches: http://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/Sketches/Datalogger.zip
Three sample sketches show how to use the SD Card and Real Time Calendar Clock to record data

- SD_Datalogger: Reads three A/Ds and records their values along with elapsed time
- SD_Datalogger_with_interval Similar program with options for frequency of recording and interpretation
- SD_Datalogger_with_RTC Similar program but with actual time instead of elapsed time

Formatting the SD Card







Most SD Cards come formatted, but it is highly suggested you re-format your card using the utility on the SDCard.org (https://www.sdcard.org/downloads/formatter_4/). Plug the card into your laptop or external reader and run the utility

Install Arduino IDE and Libraries

Working Directory Tree:
(My) Documents
Arduino
libraries
RTClib
<more…>

Note: You can skip this step if you have already installed the IDE

STEP 1: INSTALL THE ARDUINO IDE
- Download the IDE (currently named "arduino-1.0.5-r2-windows.exe") and click on it to install
- It will be installed in the Program Files directory
- Your sketches and libraries will go in the Arduino folder in (my) Documents
- Also allow the installation of the device driver software

STEP 2: INSTALL THE LIBRARIY
- Download and install the Real Time Clock library from Adafruit (note that you will need to rename the folder with the library from RTClib-master to RTClib
  https://github.com/adafruit/RTClib/archive/master.zip

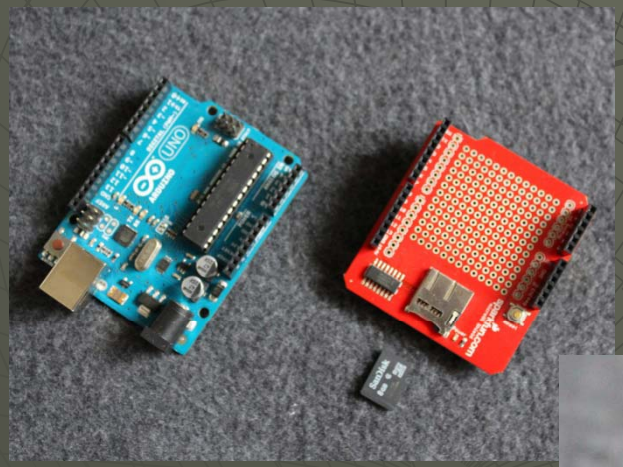
- Library install instructions are here:
  http://arduino.cc/en/Guide/Libraries#.Uy4bYU1OUpA

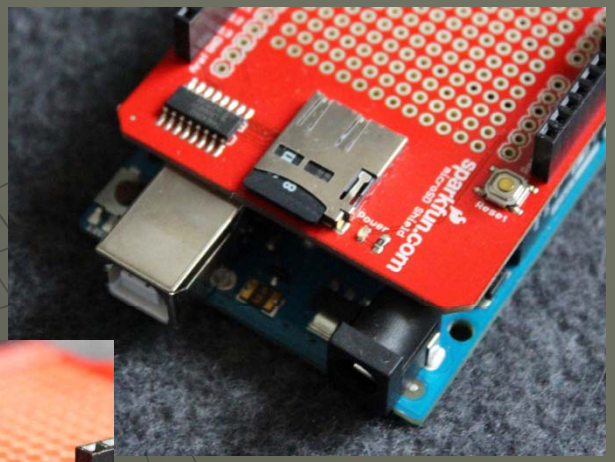
- To automatically install a downloaded .zip file library, start the Arduino IDE and click on: SKETCH->IMPORT LIBRARY->ADD LIBRARY then navigate to where the libraries were downloaded and click on the library (.zip file or folder) – check that the library has been added under "contributed" in the list under SKETCH->IMPORT LIBRARY->ADD LIBRARY

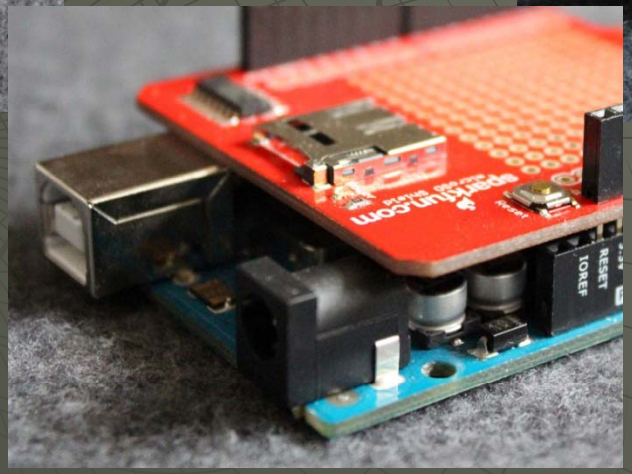
- To manually install a downloaded library, unzip it and move the folder containing all files into the "libraries" folder in the (My) Documents\Arduino\libraries, then restart the IDE

Setting up the hardware

Using shields makes wiring the hardware extremely easy – just plug the shield into the Arduino (and plug in the SD card) – then plug the USB cable in and download the sketch!

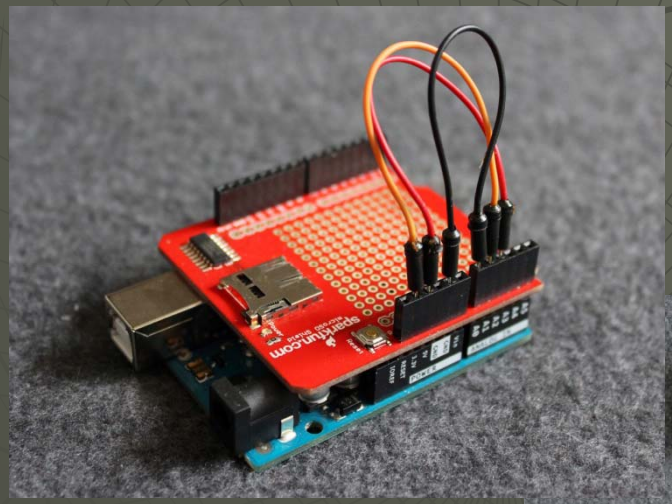


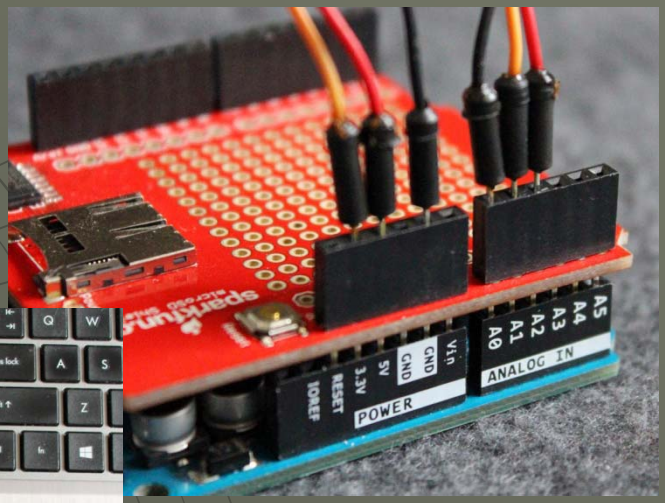




Setting up the hardware

You can run the hardware as-is, or you can attach known voltages to the A/D inputs. We can connect one A/D to GND (0 volts), one to 3.3 volts, and one to 5 volts. If you do not do this, the program will still work OK, but you will see random A/D values

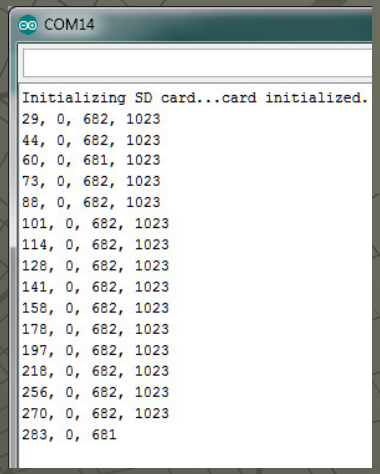


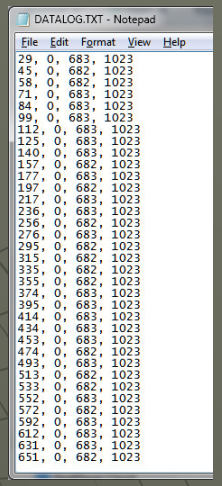


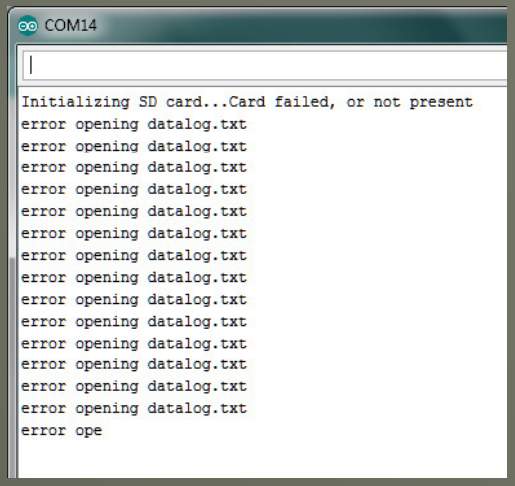
Download the LCD sketches from the AIAA OC Rocketry web sites:
Sketches: http://aiaaocrocketry.org/AIAAOCRocketryDocs/SPARC2014/Sketches/Datalogger.zip

Unzip the folders with the .ino file into the Sketches folder under Arduino in (My)Documents. The first sketch we will use is the SD_Datalogger. This sketch requires the Sparkfun micro SDcard shield and will read the A/D values, send them out the serial port as well as record them on the SD Card.

```
if (dataFile)
 {
  long timeStamp = millis();
  dataFile.print(timeStamp);
  dataFile.print(", ");
  Serial.print(timeStamp);
  Serial.print(", ");

  // read three sensors and append to the string:
  for (int analogPin = 0; analogPin < 3; analogPin++)
  {
   int sensorVal = analogRead(analogPin);
   dataFile.print(sensorVal);
   Serial.print(sensorVal);
   if (analogPin < 2)
   {
  dataFile.print(", ");
  Serial.print(", ");
   }
  }
  dataFile.println();
  dataFile.close();
  Serial.println();
 }
 else
 {
  Serial.println("error opening datalog.txt");
 }
}
```







Start the program
- Start the Serial monitor in the IDE to watch the data being read
- Remove the SD Card and plug into your reader on the PC and use notepad to read the file
- Note – the CS pin varies from shield to shield – Sparkfun uses Pin 8 – check this if you get errors

The next sketch is the SD_Datalogger_with_interval. This sketch is similar to the first with two additions

- You can set the interval between the times that the data is read.
- The program will interpret the data showing the time in seconds and the A/D readings in volts.

```
#define interpreted true
#define interval 1000
.
.
.
do  {
    lastTime = thisTime;
    thisTime = (millis() % interval);
}  while (lastTime <= thisTime);

timeStamp = millis();
File dataFile = SD.open("datalog.txt", FILE_WRITE);

// if the file is available, write to it:
if (dataFile)
{
  dataString += String(timeStamp);
#if interpreted
  dataString += " (";
  dataString += String(timeStamp/1000);
  dataString += '.';
  dataString += String (timeStamp%1000);
  dataString += "s)";
#endif
  dataString += ",";
   // read three sensors and append to the string:
  for (int analogPin = 0; analogPin < 3; analogPin++)
  {
    int sensorVal = analogRead(analogPin);
    dataString += String(sensorVal);
     .
     .
     .
```

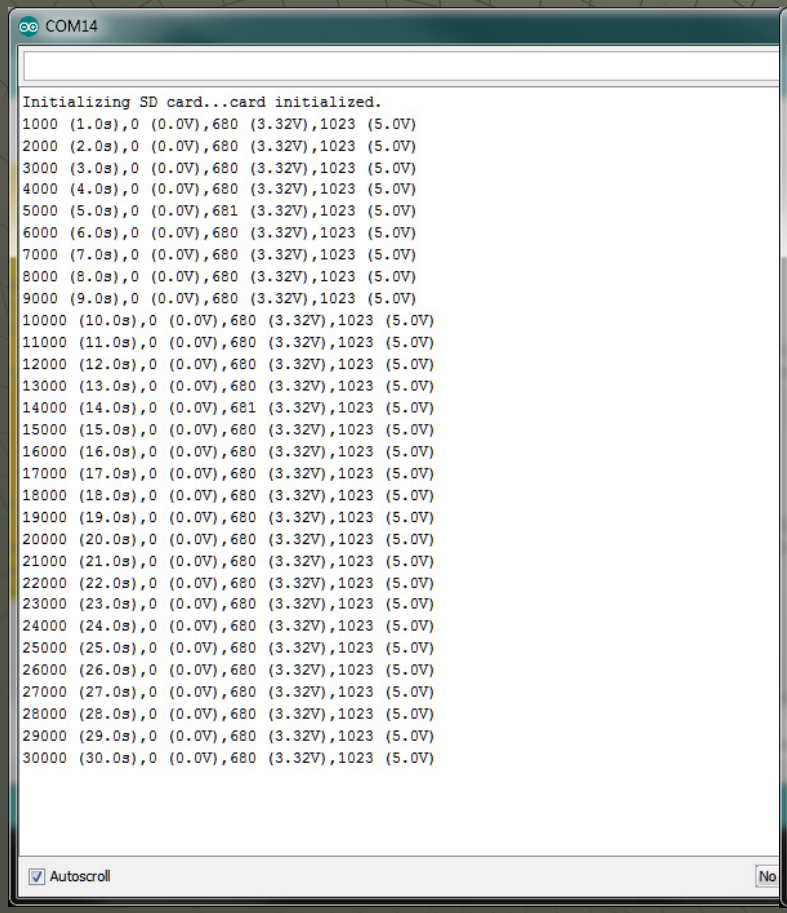
If interpreted = true seconds and volts values are included
The interval value shows how often A/Ds are read (in milliseconds)

thisTime is the new reading, lastTime is the last reading – recorded as the remainder (% divide of interval). As time progresses this will increase until it gets to the value of the interval, then it will roll back over to zero – that is our trigger

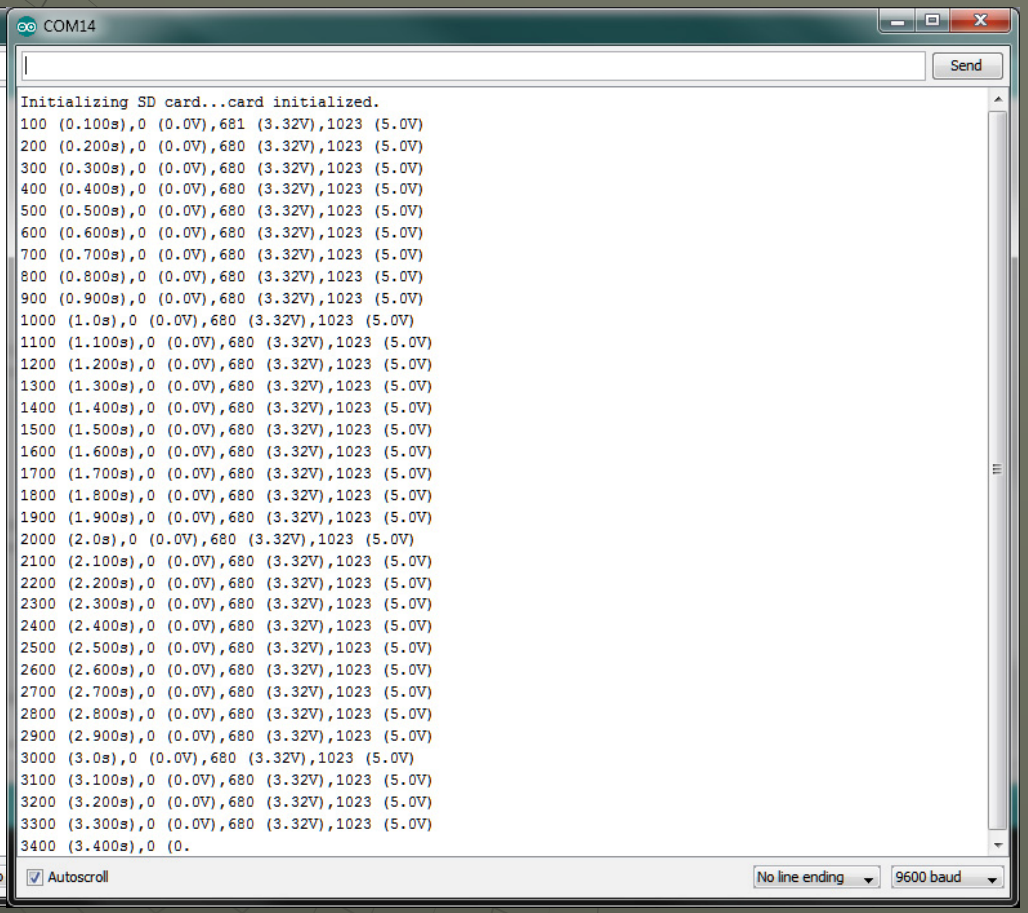
timeStamp is the time in milliseconds since the program started to execute. By dividing by 1000 we get the whole number (no remainder – this is seconds). By using modulo divide (%) of 1000 we get the reminder – the number of milliseconds without the whole seconds

Running the sketch

Interval set to 1000ms (1 second) with interpreted ON



Interval set to 100ms (1/10 second) with interpreted ON



The next sketch is the SD_Datalogger_with_interval. This sketch is similar to the first with two additions

- You can set the interval between the times that the data is read.
- The program will interpret the data showing the time in seconds and the A/D readings in volts.

```
#define interpreted true
#define interval 1000
.
.
.
 for (int analogPin = 0; analogPin < 3; analogPin++)
    {
      int sensorVal = analogRead(analogPin);
      dataString += String(sensorVal);
#if interpreted
      dataString += " (";
      sensorVal = map(sensorVal, 0, 1023, 0, 500);
      dataString += String(sensorVal / 100);
      dataString += '.';
      dataString += String(sensorVal % 100);
      dataString += "V)";
#endif
      if (analogPin < 2)
        dataString += ",";
    }
    dataString += '\r';
    dataString += '\n';
    dataFile.print(dataString);
    Serial.print(dataString);
    dataFile.close();
  }
.
.
.
```

If interpreted = true seconds and volts values are included
The interval value shows how often A/Ds are read (in milliseconds)

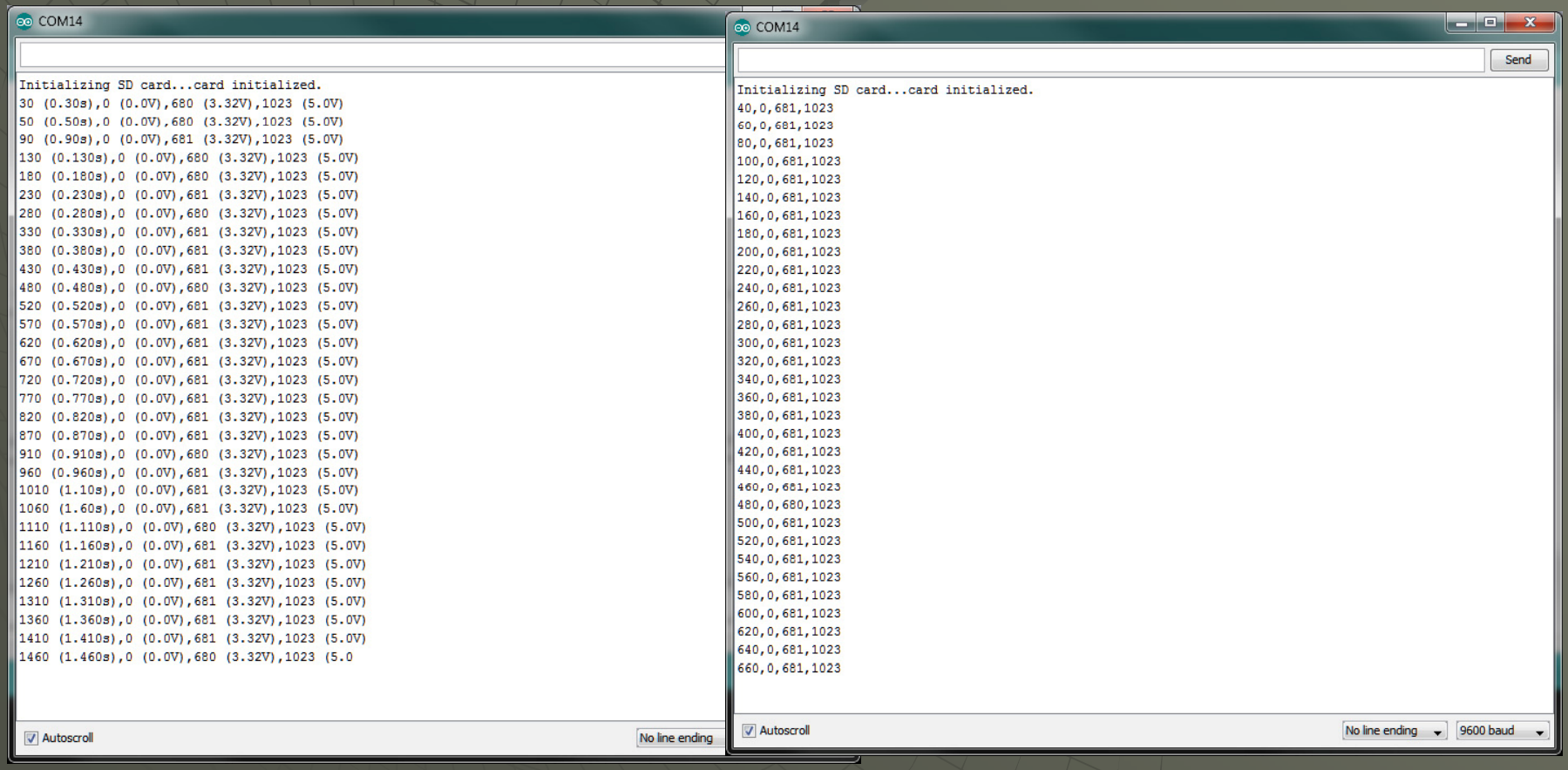
Map (sensorVal, 0, 1023, 0, 500) will map sensorVal to another value. A 10 bit A/D can count from 0 to 1023, so the first value is the minimum and the second value is the maximum value we will see in sensorVal. The second two values are the mapped values – 0 -> 0 and 1023 -> 5 volts in 1/100ths of a volt, or 500.

Similar to the seconds value, we divide (“/”) by 100 to extract full volts, and use modulo (“%”) to extract the remainder, which in this case is 1/100ths of a volt

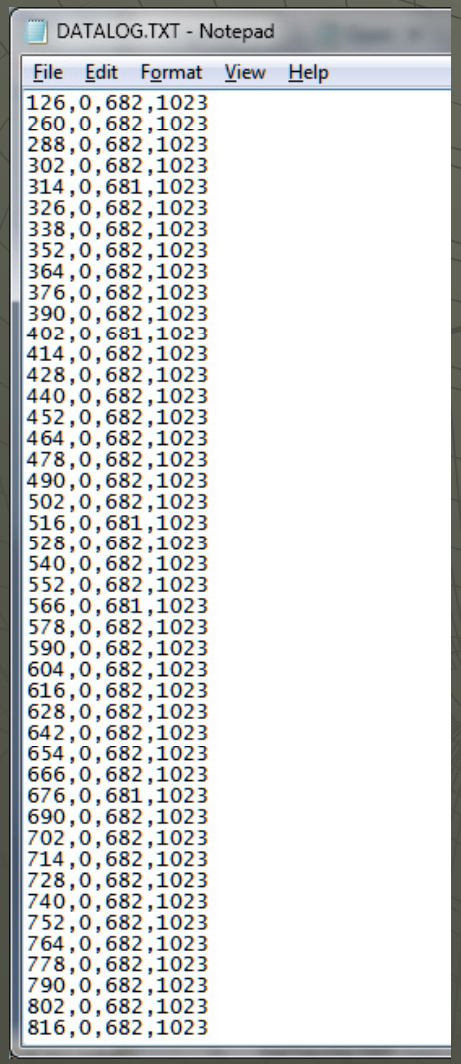
Running the sketch

Interval set to 10ms with interpreted ON – there is not enough time to read, interpret, write, and send that often so the Arduino does the best it can. At 9600 BAUD it takes about 1 ms to send 1 character

Interval set to 10ms with interpreted OFF – still not enough time but the reports are more consistent – but no interpreted values



Running the sketch



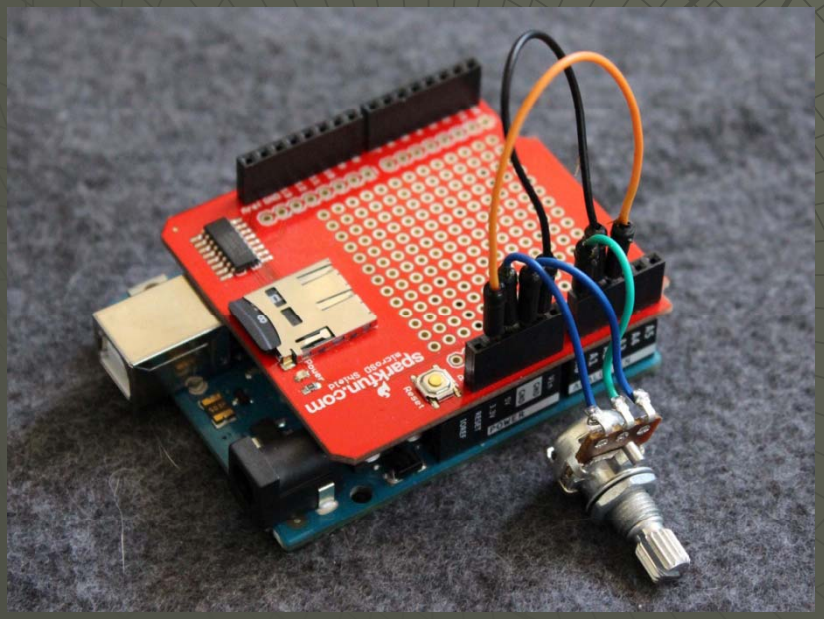
To capture data as fast as we can, we'll set the interval low ("2") and turn the serial data OFF. When we look at the data captured on the SD card, the intervals between captured data are lower – most are around 12 – 15ms – still no where near the 2ms that we asked for.

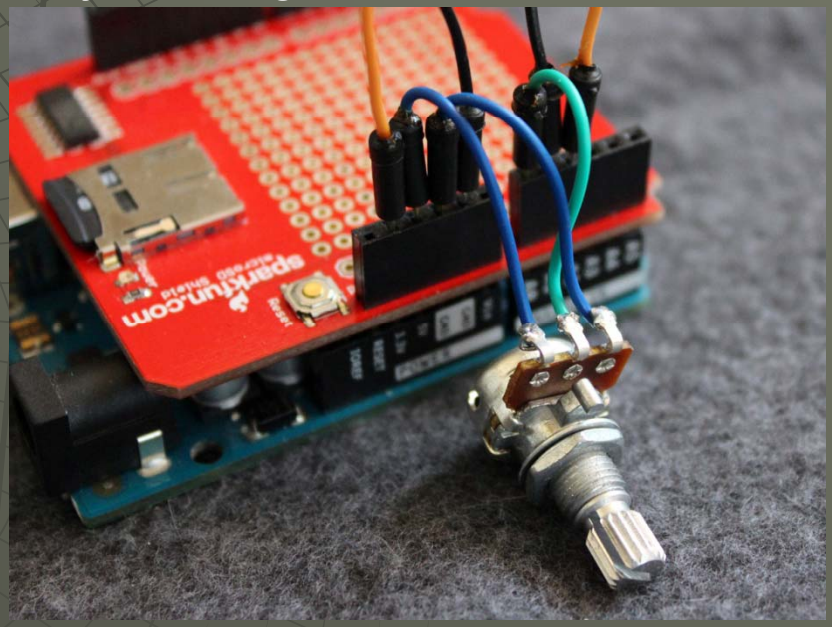
Be aware of times and what you are asking the Arduino to do when capturing data and be realistic

To minimize the amount of data captured, and therefore the size of the file, you may want to add a "G" switch that closes when you sense the rocket taking off (2.1Gs) - available from Perfectflite at: http://www.perfectflite.com/gs21.html - start capturing data when the switch closes. Or you can trigger on the data from your accelerometer if you have one in your payload

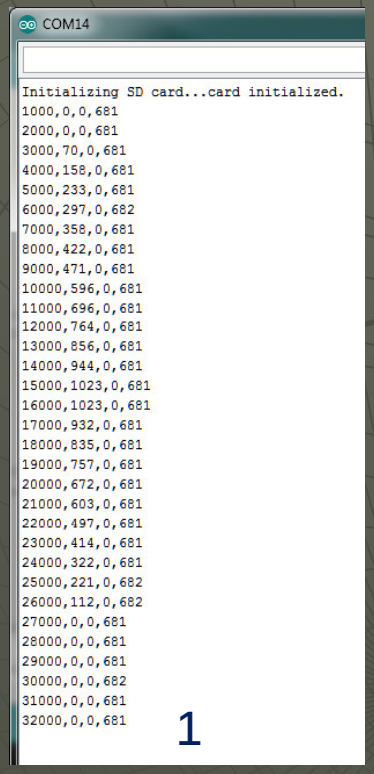
Graphing your data

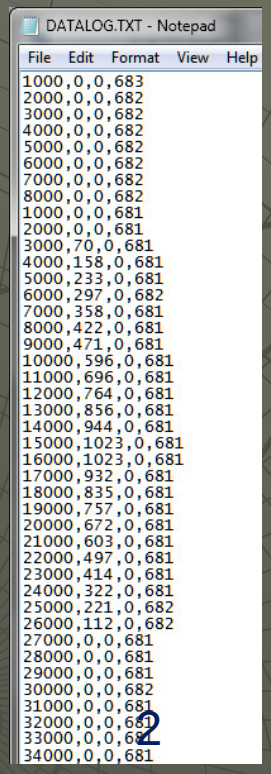
Graphs of flat lines – which is what you get when you attach an A/D to a constant voltage like 0, 3.3V, or 5V is not very interesting. You can use the same sketch but attach one of the A/Ds to a potentiometer – one side of the "pot" is connected to 5VDC and the other side to GND. The center is connected to the A/D input (we used AD0) – with AD1 connected to GND and AD2 connected to 3.3V. Vary the voltage by turning the shaft.

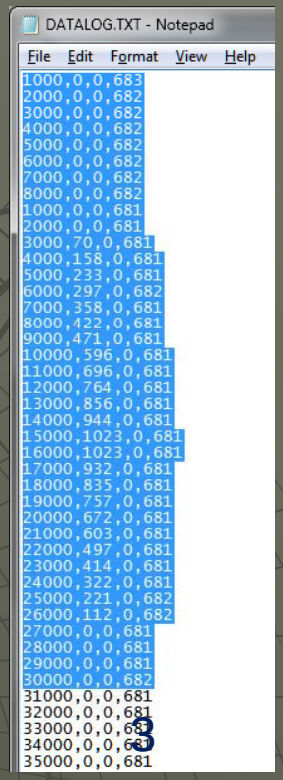


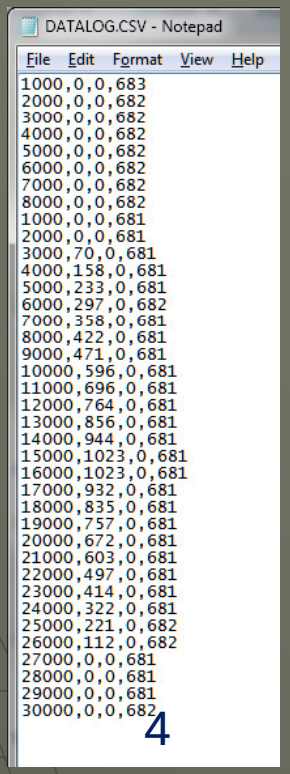


Graphing your data



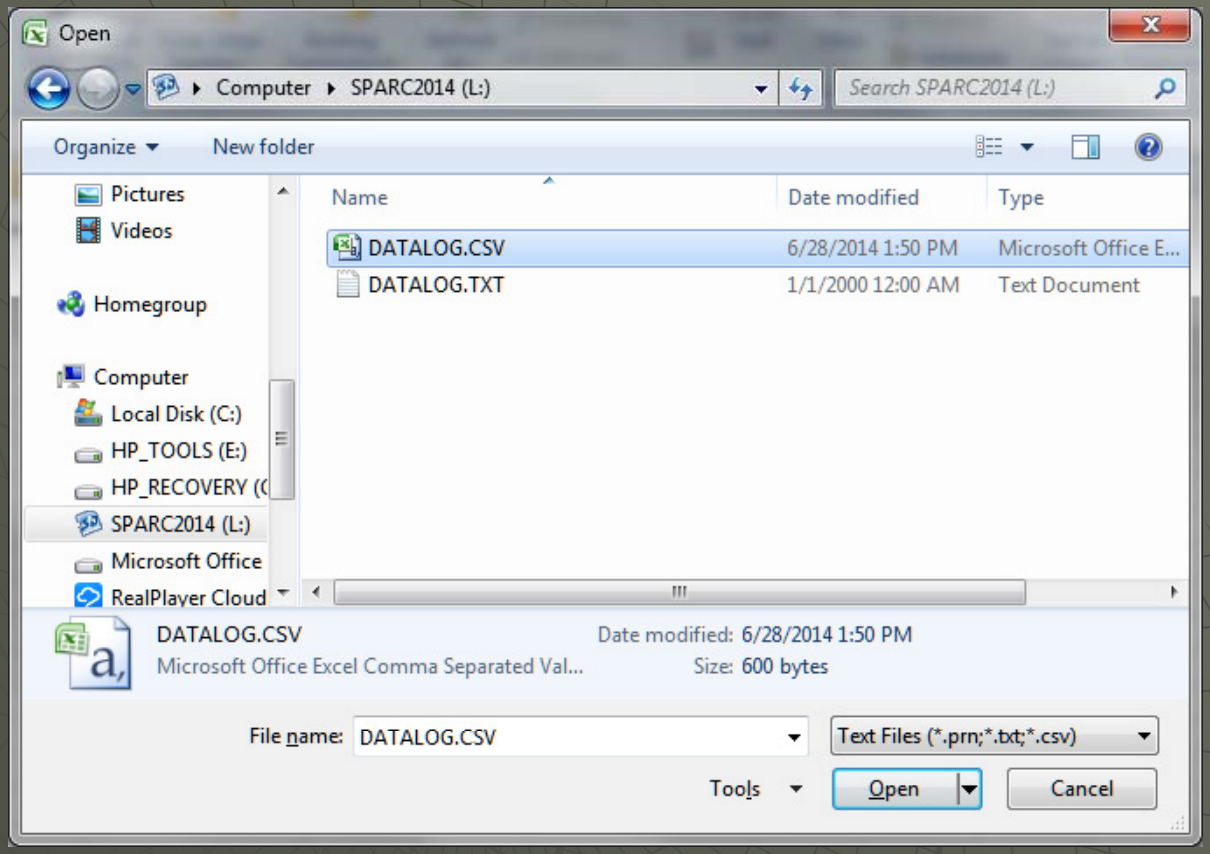


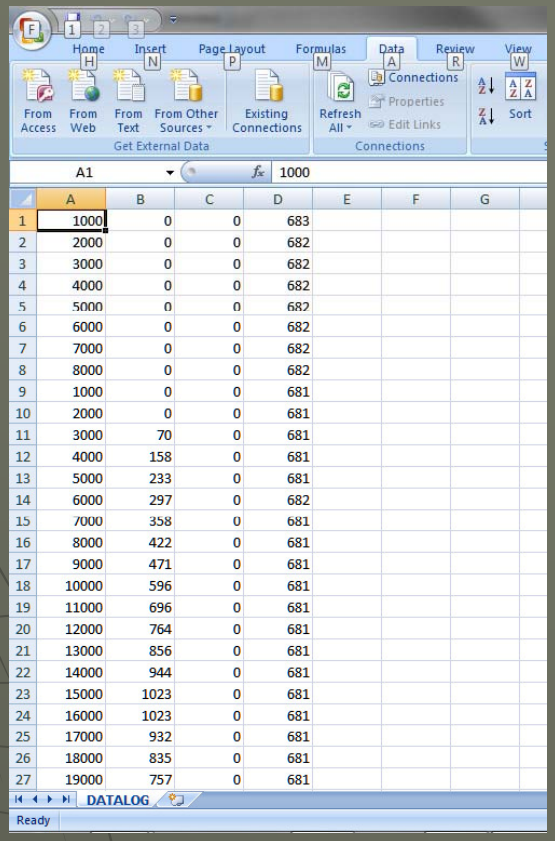




1 – Data from the serial COMM port as the voltage is varied
2 – Data captured in the DATALOG.TXT file on the SD Card
3 – Select the data you want to graph
4 – Copy it into a new file and save as DATALOG.CSV
Note: CSV is Comma Separated Values – between each value is ","

Graphing your data

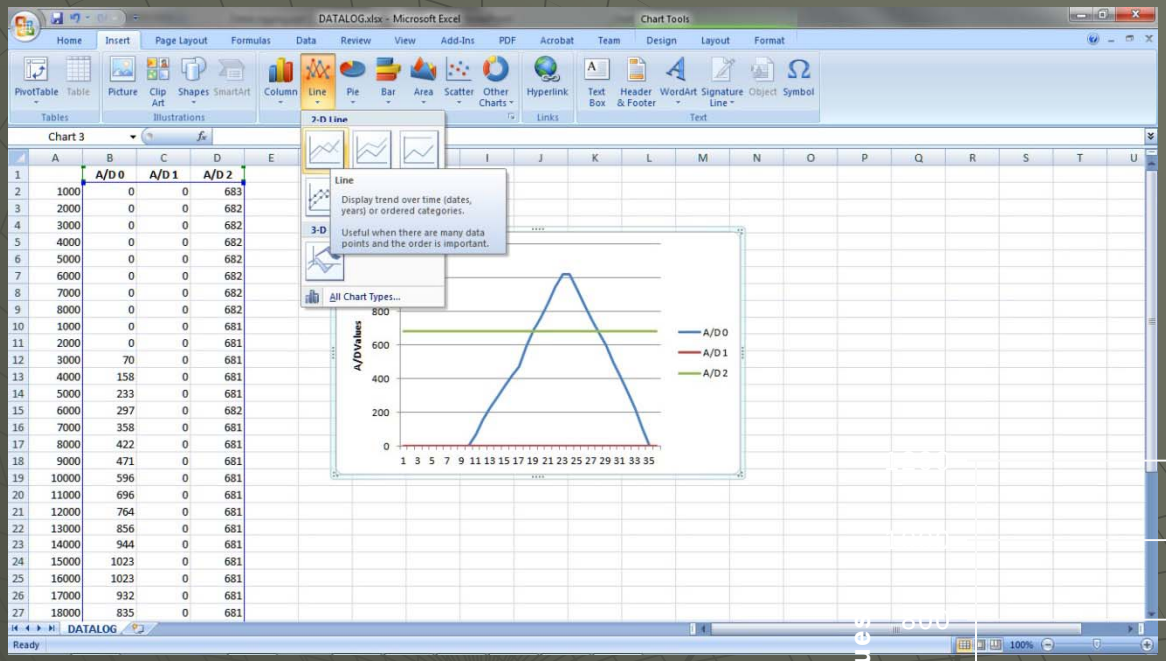




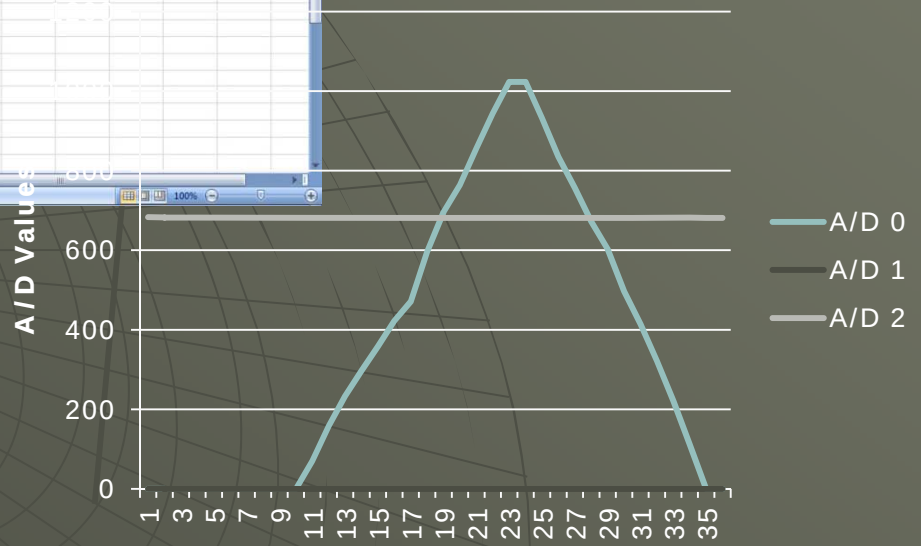
	A	B	C	D
1	1000	0	0	683
2	2000	0	0	682
3	3000	0	0	682
4	4000	0	0	682
5	5000	0	0	682
6	6000	0	0	682
7	7000	0	0	682
8	8000	0	0	682
9	1000	0	0	681
10	2000	0	0	681
11	3000	70	0	681
12	4000	158	0	681
13	5000	233	0	681
14	6000	297	0	682
15	7000	358	0	681
16	8000	422	0	681
17	9000	471	0	681
18	10000	596	0	681
19	11000	696	0	681
20	12000	764	0	681
21	13000	856	0	681
22	14000	944	0	681
23	15000	1023	0	681
24	16000	1023	0	681
25	17000	932	0	681
26	18000	835	0	681
27	19000	757	0	681

Start Microsoft Excel (or similar from Open Office) and open the DATALOG.CSV file – each value separated by a comma in the .csv file should be in its own cell in Excel – each row is a new time and each column should represent one of the A/Ds

Graphing your data



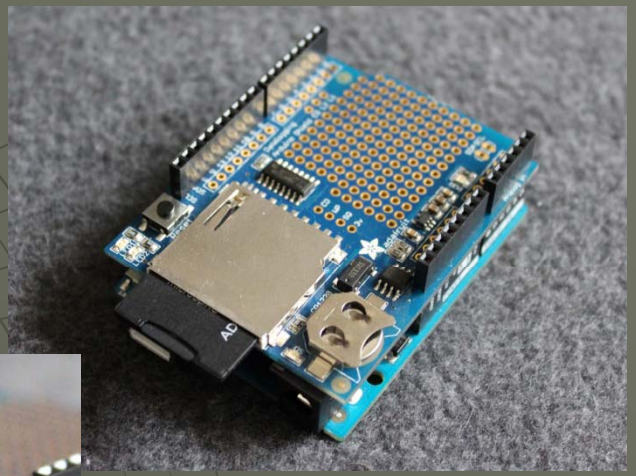
Add names for each line in the graph at the top of the column then select the data to graph by selecting columns including the names for titles

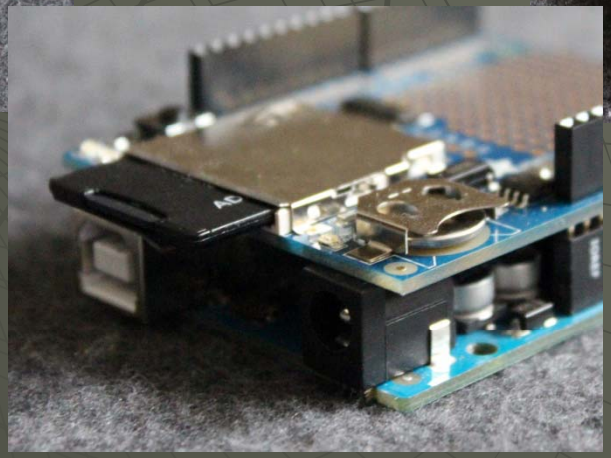


Click on Insert, then the line chart – this will create a chart in your spreadsheet. You can copy that chart to another document

Setting up the hardware

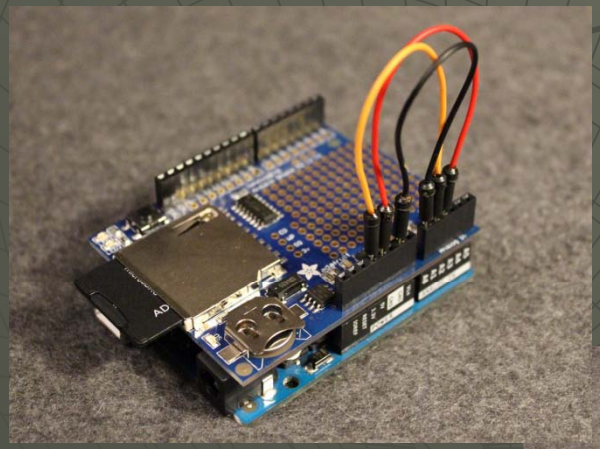
If you want to record the actual date and time instead of the time interval since the program started, you'll need a Real Time Calendar/Clock (RTC). The Adafruit shield provides the RTC as well as an interface for the SD Card (full sized – you'll need an adapter for the micro SD card

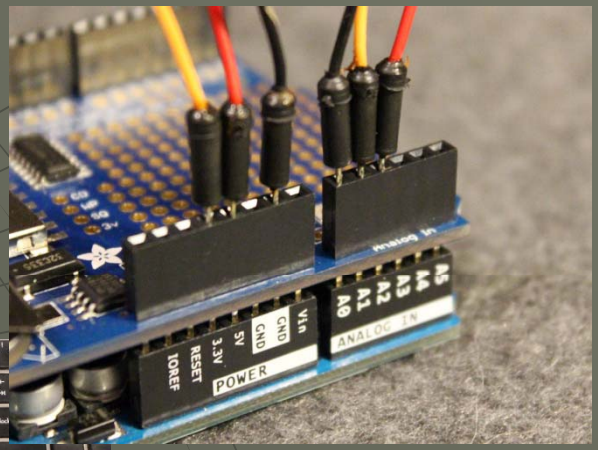


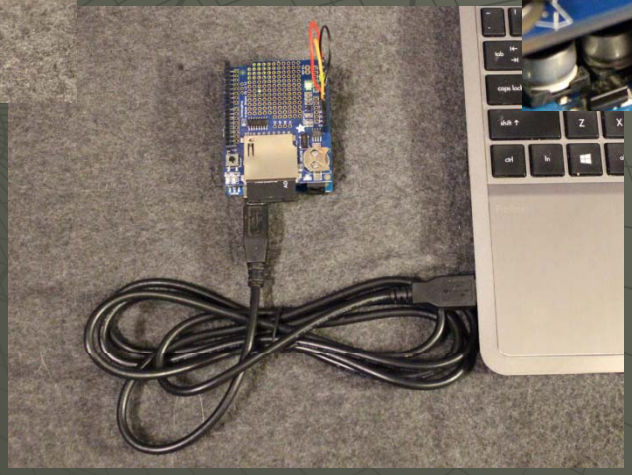


Setting up the hardware

Once again, you can run the hardware as-is, or you can attach known voltages to the A/D inputs. We can connect one A/D to GND (0 volts), one to 3.3 volts, and one to 5 volts. If you do not do this, the program will still work OK, but you will see random A/D values

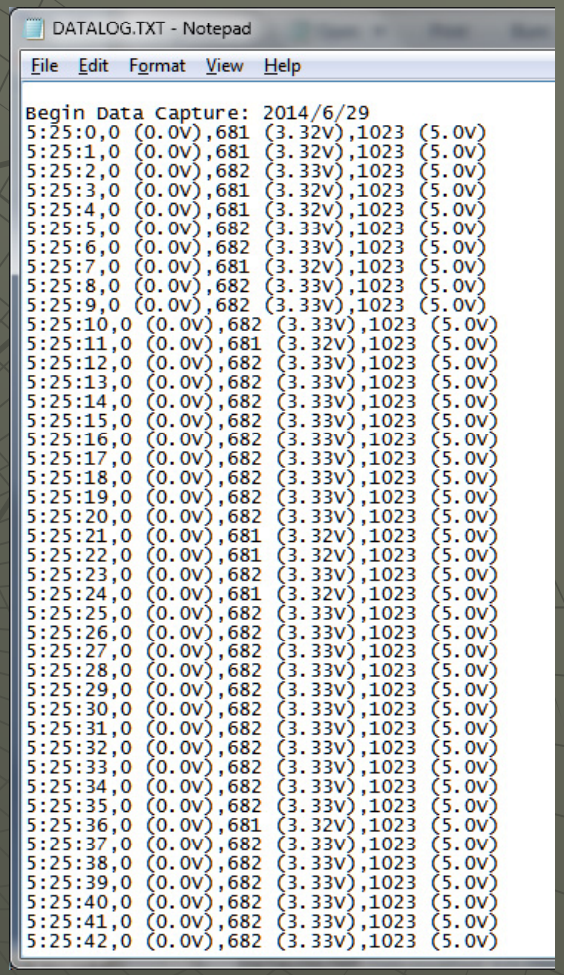






For the RTC hardware we'll use the SD_Datalogger._with_RTC. This sketch requires the Adafruit micro SDcard shield with RTC and will read the A/D values, send them out the serial port as well as record them on the SD Card. But instead of seeing the time interval since the program started, we'll see the actual date and time the data was captured. Note that this program requires the RTClib library. When compiled the program grabs the date and time from your computer, then when the program runs, the RTC is set.

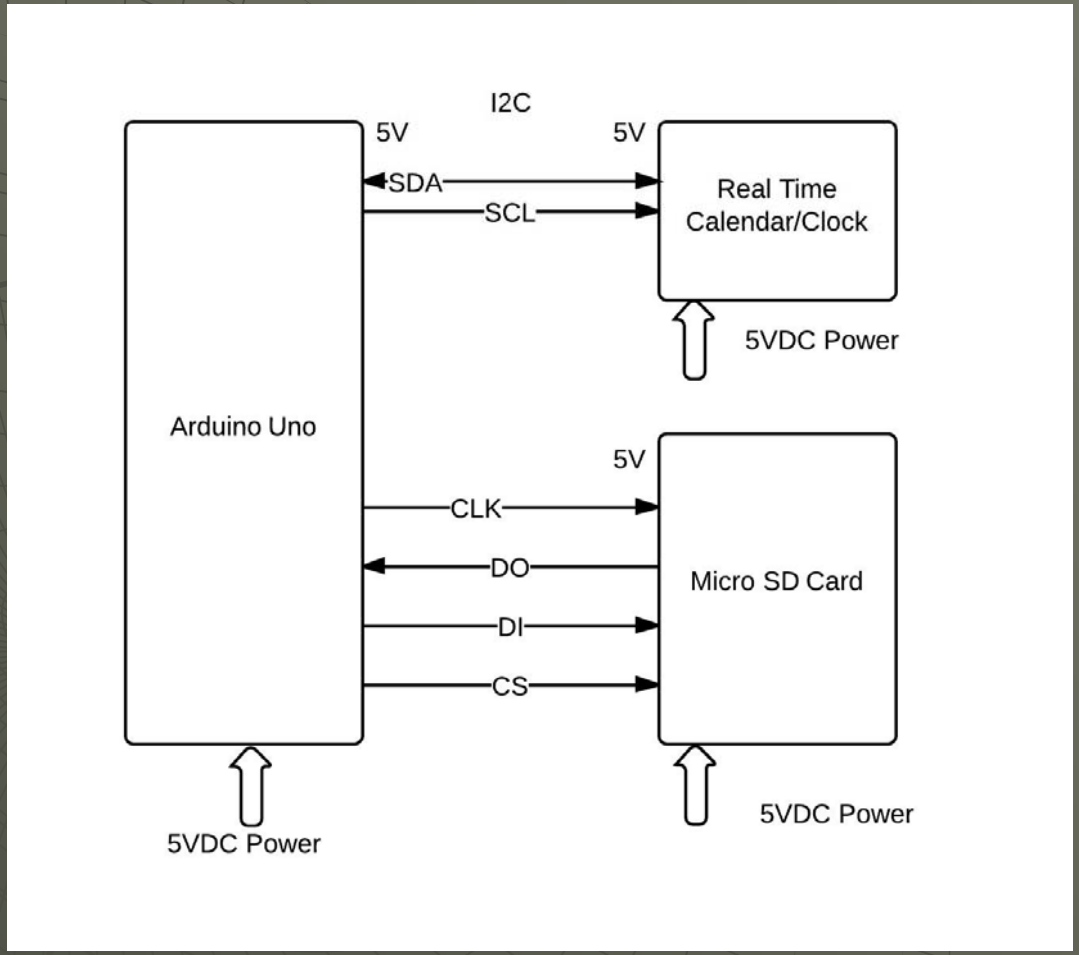
```
if (dataFile)      {

    DateTime now = rtc.now();
    dataString += String(now.hour(), DEC);
    dataString += (':');
    dataString += String(now.minute(), DEC);
    dataString += (':');
    dataString += String(now.second(), DEC);

    dataString += ",";

    for (int analogPin = 0; analogPin < 3; analogPin++)
    {
      int sensorVal = analogRead(analogPin);
      dataString += String(sensorVal);
#if interpreted
      dataString += " (";
      sensorVal = map(sensorVal, 0, 1023, 0, 500);
      dataString += String(sensorVal / 100);
      dataString += '.';
      dataString += String(sensorVal % 100);
      dataString += "V)";
#endif
      if (analogPin < 2)
        dataString += ",";
    }
    dataString += '\r';
    dataString += '\n';
    dataFile.print(dataString);
    Serial.print(dataString);
    dataFile.close();
  }
```



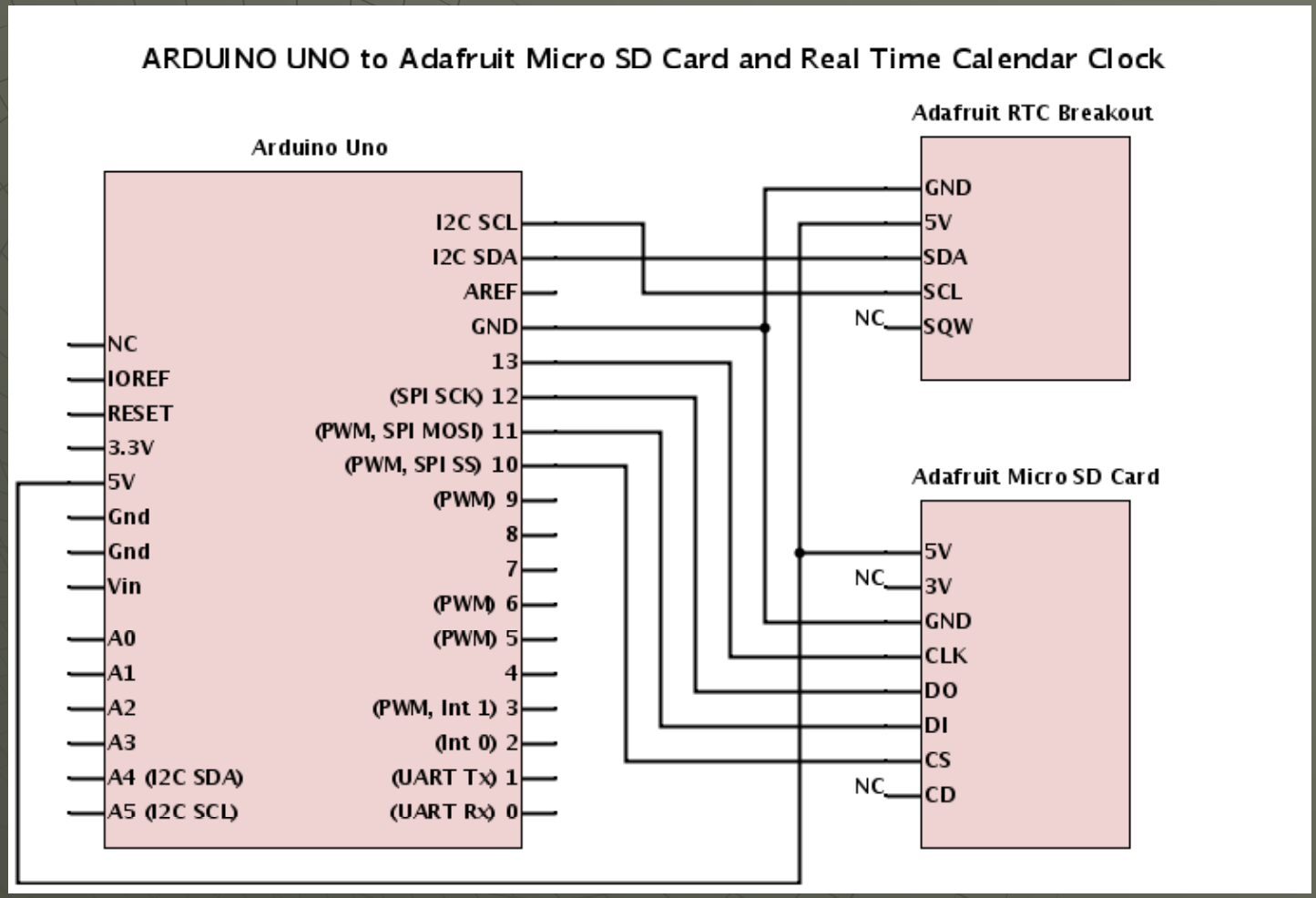
Start the program – you have similar options for interval of recording data and interpreted or not

- Start the Serial monitor in the IDE to watch the data being read
- Remove the SD Card and plug into your reader on the PC and use notepad to read the file
- Note – the CS pin varies from shield to shield – Adafruit uses Pin 10 – check this if you get errors

Connecting the Uno – Micro SD Card - RTC

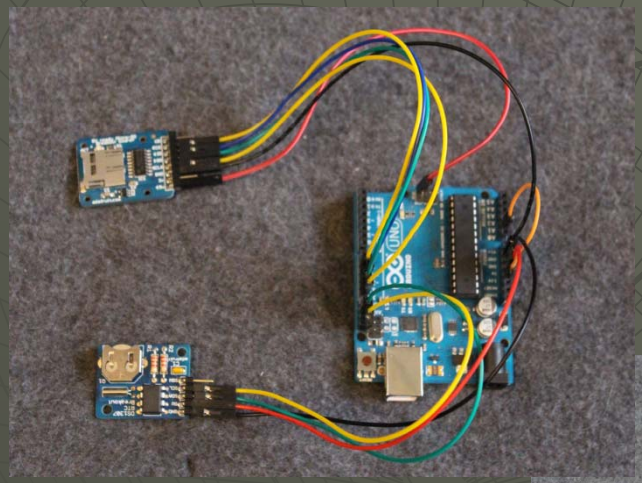


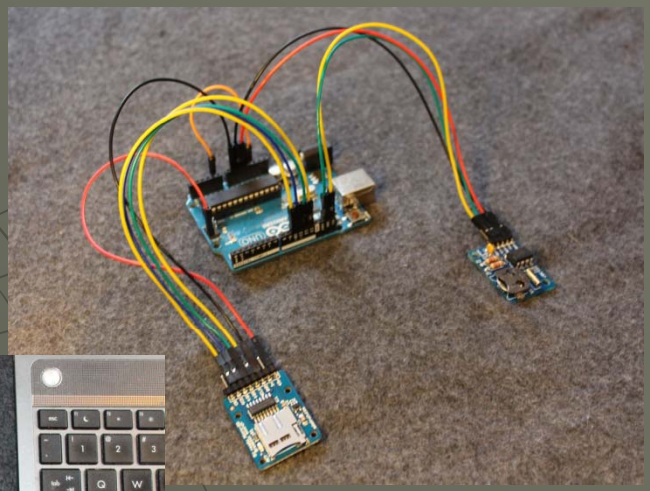
Connecting the Uno – Micro SD Card - RTC

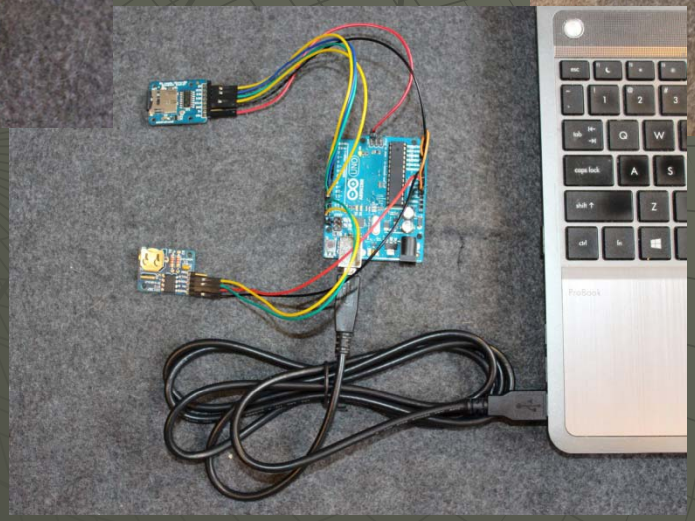


Setting up the discrete hardware

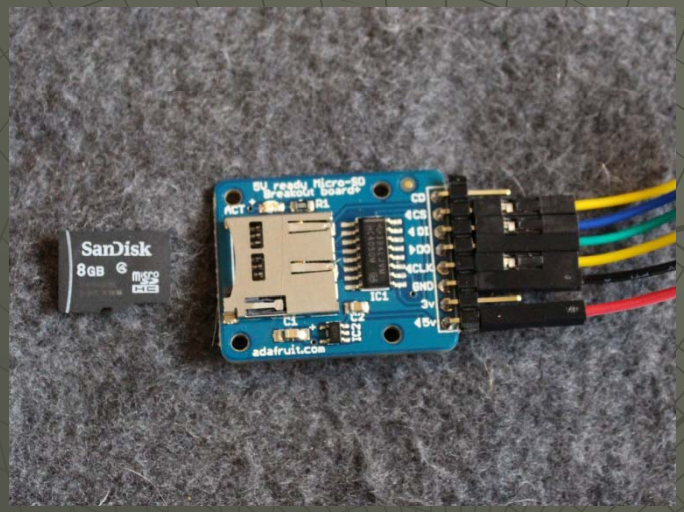
In your final flight hardware, using a shield might not be the best – or there may be no room. You can do the same thing as the shield with individual break-out boards for the SD Card and the Real Time clock.

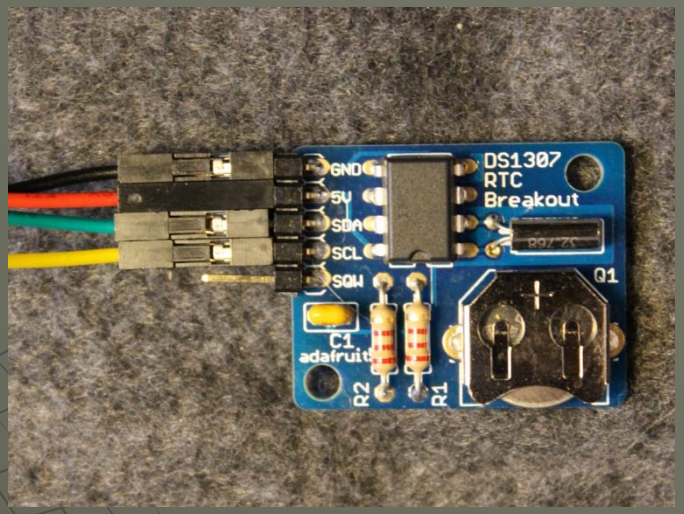


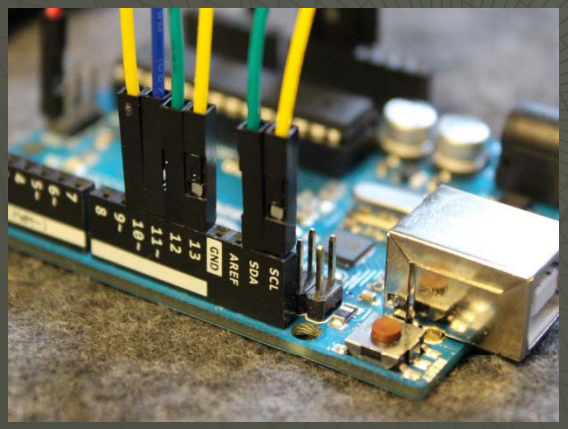


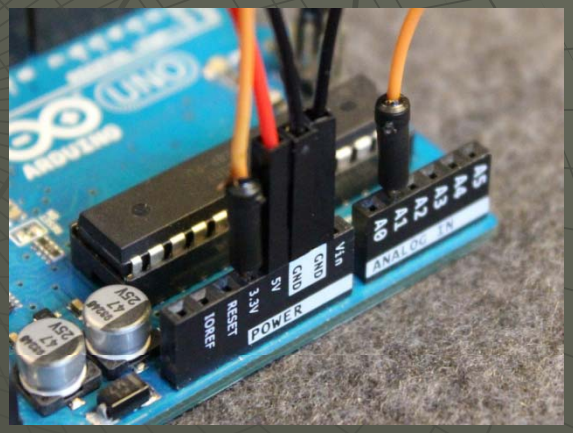


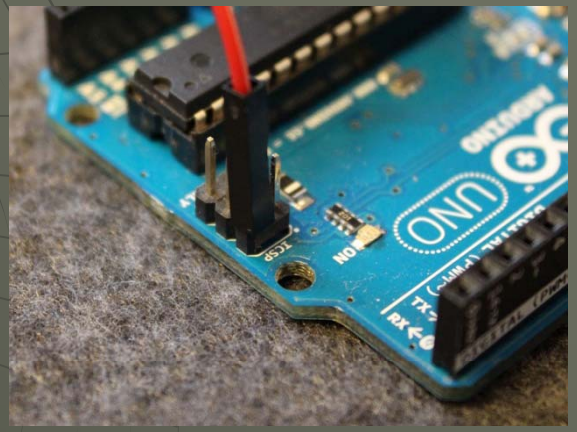
Setting up the discrete hardware



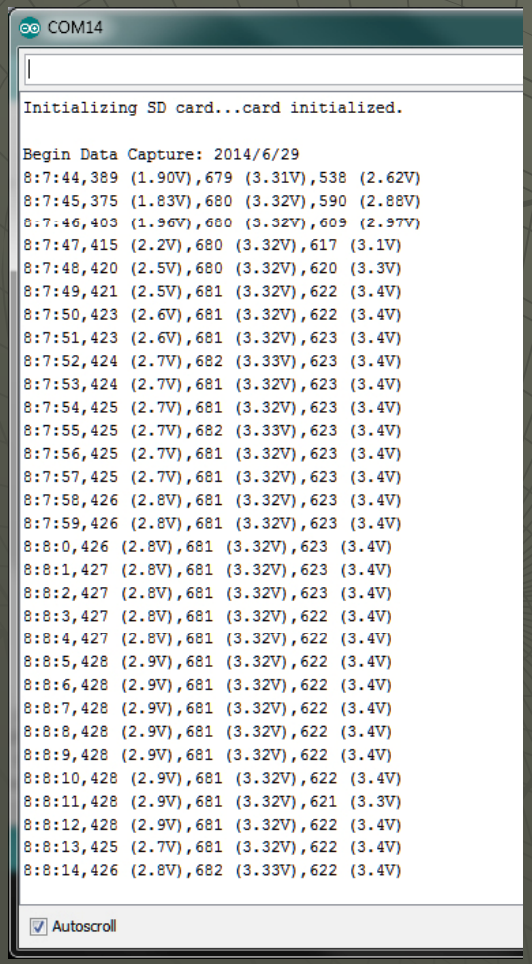


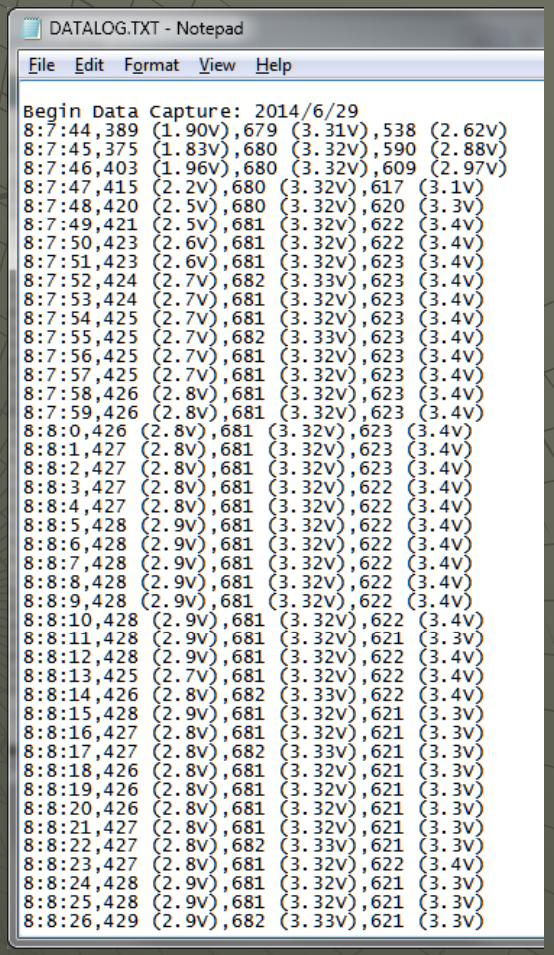






For the RTC breakout board hardware we'll use the same SD_Datalogger._with_RTC with no changes. If thi I the first time you are running the board, make certain to recompile the sketch and load it before connecting the Real Time Clock since this will set your clock. If you need to reset the time, remove the battery and disconnect the RTC, recompile and load. Disconnect the power from the Arduino, reconnect the RTC and start power the Arduino again.





Start the program – you have similar options for interval of recording data and interpreted or not

- Start the Serial monitor in the IDE to watch the data being read
- Remove the SD Card and plug into your reader on the PC and use notepad to read the file
- Note – the CS pin varies from shield to shield – Adafruit uses Pin 10 – if you followed the schematic and wiring photos you should be correct

APPENDIX I

Hardware and Libraries

In this appendix is background information on hardware and the libraries that control that hardware. You don't need to completely understand this information, but you do need to understand how to use the libraries

SD Library I

The following routines make up the LCD SD Card library – you will be using these:

SD Class (accessing the SD card and manipulating files and directories)

begin() – initializes the SD Library and card
http://arduino.cc/en/Reference/SDbegin

exists() – tests whether a file or directory exists on the SD card
http://arduino.cc/en/Reference/SDexists

mkdir() – creates a directory on the SD card
http://arduino.cc/en/Reference/SDmkdir

open() – opens a file on the SD card
http://arduino.cc/en/Reference/SDopen

remove() – remove a file from the SD card
http://arduino.cc/en/Reference/SDremove

rmdir() – remove a directory from the SD card
http://arduino.cc/en/Reference/SDrmdir

SD Library II

File Class (allow for reading from and writing to files on an SD Card)

available() – check if there are any bytes available for reading from the file
http://arduino.cc/en/Reference/FileAvailable

close() – close the file and assure any written data is physically saved to the card
http://arduino.cc/en/Reference/FileClose

flush() – ensures any bytes written to the file are physically saved to the card
http://arduino.cc/en/Reference/FileFlush

peek() – read a byte from a file without advancing
http://arduino.cc/en/Reference/FilePeek

position() – get the current file position
http://arduino.cc/en/Reference/FilePosition

print() – print data to already opened file as sequence of ASCII chars (e.g. 123 is '1', '2', '3')
http://arduino.cc/en/Reference/FilePrint

Println() – print data as above followed by CR and LF for a new line
http://arduino.cc/en/Reference/FilePrintln

SD Library III

seek() – seek a new position in the file between 0 and the size of the file (inclusive)
http://arduino.cc/en/Reference/FileSeek

size() – get the size of a file
http://arduino.cc/en/Reference/FileSize

read() – read a byte from a file
http://arduino.cc/en/Reference/FileRead

write() – write data to a file
http://arduino.cc/en/Reference/FileWrite

isDirectory() – reports if the current file is a directory or not
http://arduino.cc/en/Reference/FileIsDirectory

openNextFile() – reports the next file or folder in a directory
http://arduino.cc/en/Reference/FileOpenNextFile

rewindDirectory() – brings you back to the first file in a directory (used with openNextFile())
http://arduino.cc/en/Reference/FileRewindDirectory