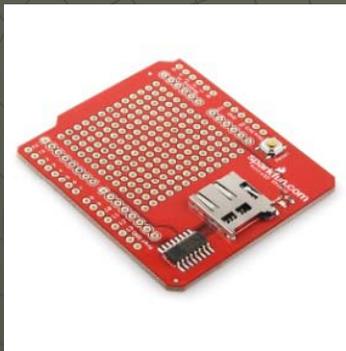


Hardware Required

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

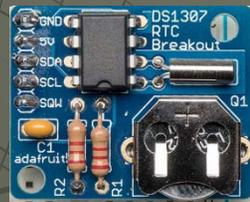
-SD Card Shield with Real Time Calendar Clock:

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

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

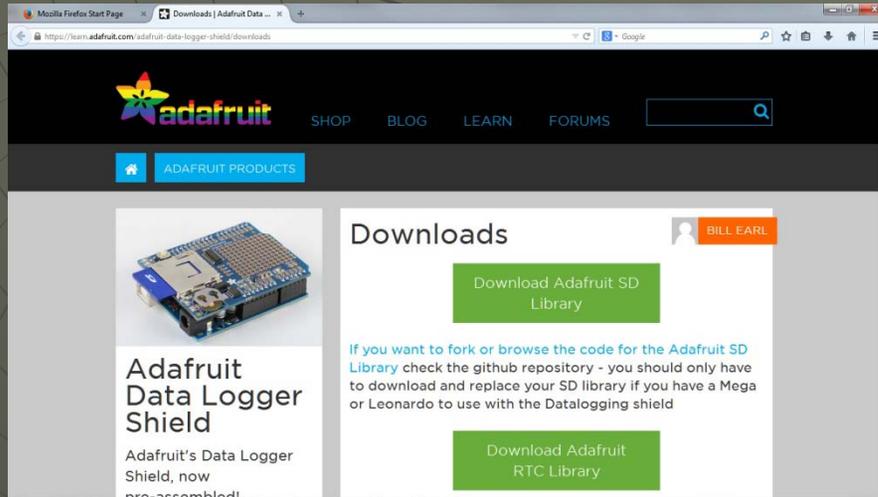
- Purchase: <http://www.amazon.com/Arduino-UNO-board-DIP-ATmega328P/dp/B006H06TVG>
- Schematic: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf

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#.Uy4WgU1QUpA>

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)

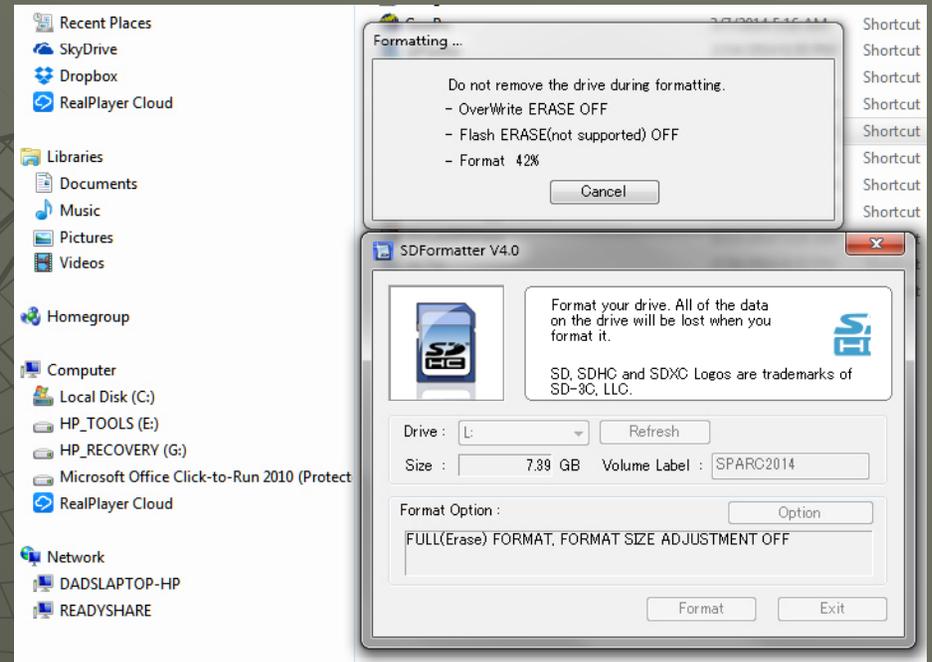
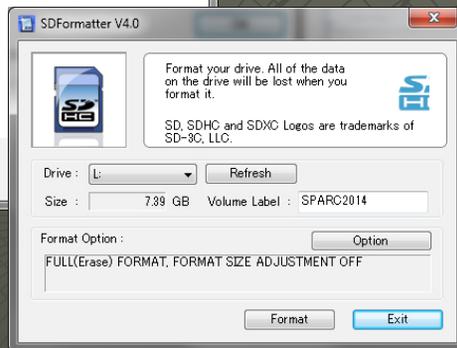
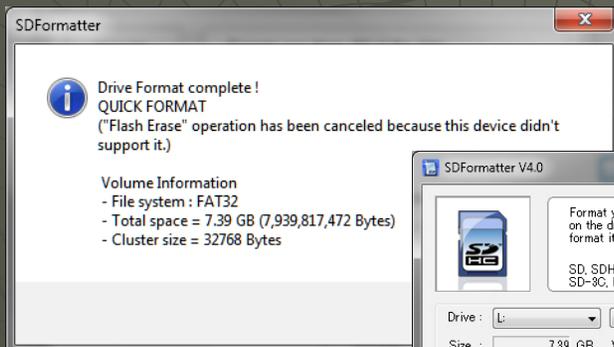
<https://github.com/adafruit/RTClib/archive/master.zip>

Sketches: <http://aiaacrocketry.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

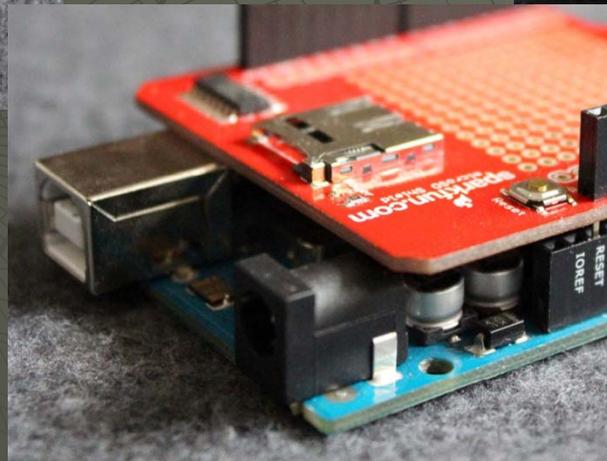
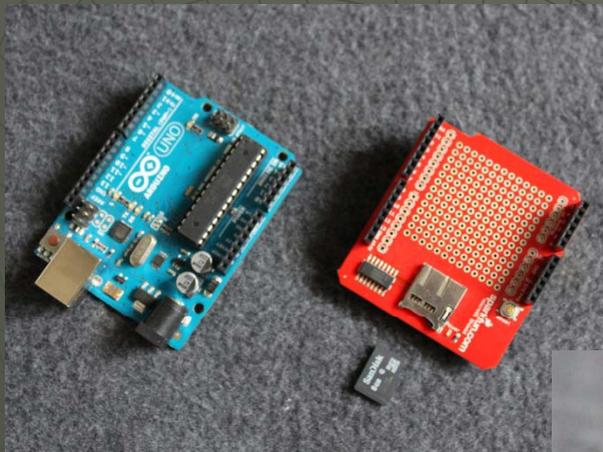
STEP 2: INSTALL THE LIBRARY

- 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

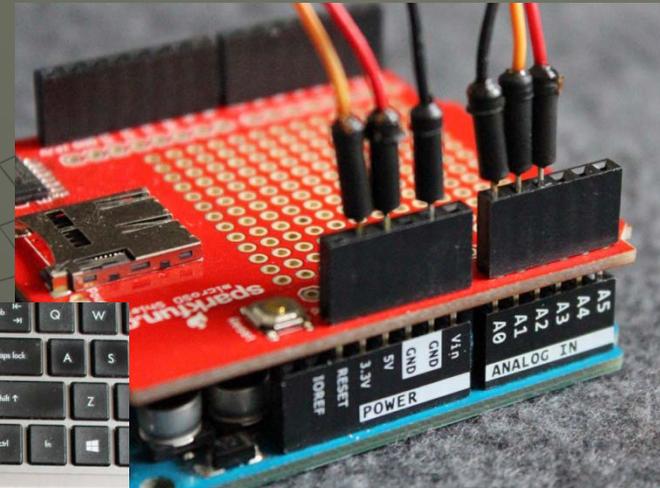
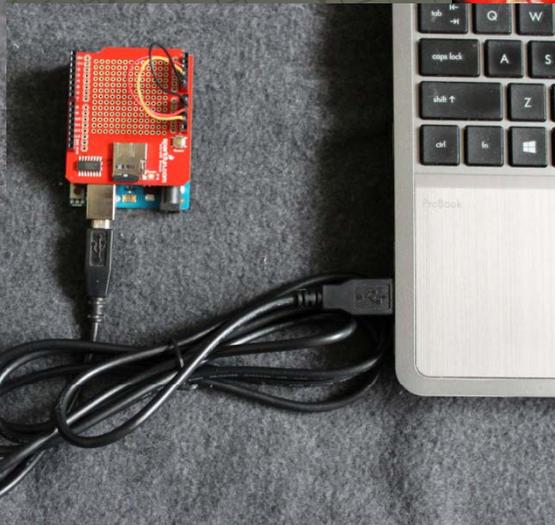
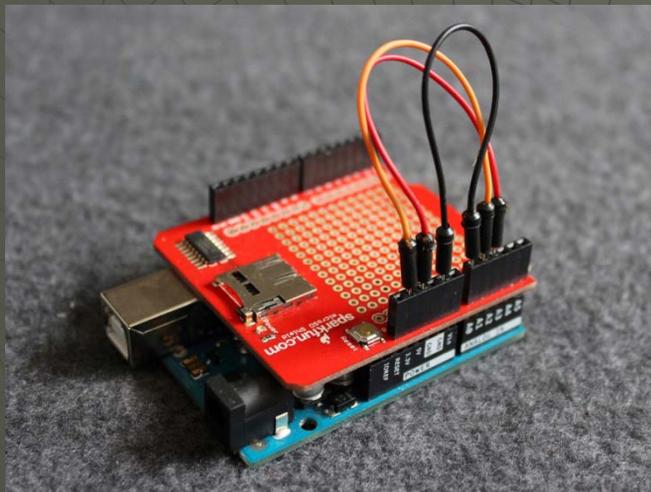
Using shields makes wiring the hardware extremely easy – just plug the shield into the Arduino (and plug in the SD card) – then plug the USB cable in and download the sketch!





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





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
```

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

```
do {  
  lastTime = thisTime;  
  thisTime = (millis() % interval);  
} while (lastTime <= thisTime);
```

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 = 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
```

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

```
dataString += ",";  
  // read three sensors and append to the string:  
  for (int analogPin = 0; analogPin < 3; analogPin++)  
  {  
    int sensorVal = analogRead(analogPin);  
    dataString += String(sensorVal);
```



Running the sketch

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

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

```
COM14
Initializing SD card...card initialized.
1000 (1.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
2000 (2.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
3000 (3.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
4000 (4.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
5000 (5.0s),0 (0.0V),681 (3.32V),1023 (5.0V)
6000 (6.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
7000 (7.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
8000 (8.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
9000 (9.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
10000 (10.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
11000 (11.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
12000 (12.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
13000 (13.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
14000 (14.0s),0 (0.0V),681 (3.32V),1023 (5.0V)
15000 (15.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
16000 (16.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
17000 (17.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
18000 (18.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
19000 (19.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
20000 (20.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
21000 (21.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
22000 (22.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
23000 (23.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
24000 (24.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
25000 (25.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
26000 (26.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
27000 (27.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
28000 (28.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
29000 (29.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
30000 (30.0s),0 (0.0V),680 (3.32V),1023 (5.0V)

COM14
Initializing SD card...card initialized.
100 (0.100s),0 (0.0V),681 (3.32V),1023 (5.0V)
200 (0.200s),0 (0.0V),680 (3.32V),1023 (5.0V)
300 (0.300s),0 (0.0V),680 (3.32V),1023 (5.0V)
400 (0.400s),0 (0.0V),680 (3.32V),1023 (5.0V)
500 (0.500s),0 (0.0V),680 (3.32V),1023 (5.0V)
600 (0.600s),0 (0.0V),680 (3.32V),1023 (5.0V)
700 (0.700s),0 (0.0V),680 (3.32V),1023 (5.0V)
800 (0.800s),0 (0.0V),680 (3.32V),1023 (5.0V)
900 (0.900s),0 (0.0V),680 (3.32V),1023 (5.0V)
1000 (1.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
1100 (1.100s),0 (0.0V),680 (3.32V),1023 (5.0V)
1200 (1.200s),0 (0.0V),680 (3.32V),1023 (5.0V)
1300 (1.300s),0 (0.0V),680 (3.32V),1023 (5.0V)
1400 (1.400s),0 (0.0V),680 (3.32V),1023 (5.0V)
1500 (1.500s),0 (0.0V),680 (3.32V),1023 (5.0V)
1600 (1.600s),0 (0.0V),680 (3.32V),1023 (5.0V)
1700 (1.700s),0 (0.0V),680 (3.32V),1023 (5.0V)
1800 (1.800s),0 (0.0V),680 (3.32V),1023 (5.0V)
1900 (1.900s),0 (0.0V),680 (3.32V),1023 (5.0V)
2000 (2.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
2100 (2.100s),0 (0.0V),680 (3.32V),1023 (5.0V)
2200 (2.200s),0 (0.0V),680 (3.32V),1023 (5.0V)
2300 (2.300s),0 (0.0V),680 (3.32V),1023 (5.0V)
2400 (2.400s),0 (0.0V),680 (3.32V),1023 (5.0V)
2500 (2.500s),0 (0.0V),680 (3.32V),1023 (5.0V)
2600 (2.600s),0 (0.0V),680 (3.32V),1023 (5.0V)
2700 (2.700s),0 (0.0V),680 (3.32V),1023 (5.0V)
2800 (2.800s),0 (0.0V),680 (3.32V),1023 (5.0V)
2900 (2.900s),0 (0.0V),680 (3.32V),1023 (5.0V)
3000 (3.0s),0 (0.0V),680 (3.32V),1023 (5.0V)
3100 (3.100s),0 (0.0V),680 (3.32V),1023 (5.0V)
3200 (3.200s),0 (0.0V),680 (3.32V),1023 (5.0V)
3300 (3.300s),0 (0.0V),680 (3.32V),1023 (5.0V)
3400 (3.400s),0 (0.0V),680 (3.32V),1023 (5.0V)
```



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

```
COM14
Initializing SD card...card initialized.
30 (0.30s),0 (0.0V),680 (3.32V),1023 (5.0V)
50 (0.50s),0 (0.0V),680 (3.32V),1023 (5.0V)
90 (0.90s),0 (0.0V),681 (3.32V),1023 (5.0V)
130 (0.130s),0 (0.0V),680 (3.32V),1023 (5.0V)
180 (0.180s),0 (0.0V),680 (3.32V),1023 (5.0V)
230 (0.230s),0 (0.0V),681 (3.32V),1023 (5.0V)
280 (0.280s),0 (0.0V),680 (3.32V),1023 (5.0V)
330 (0.330s),0 (0.0V),681 (3.32V),1023 (5.0V)
380 (0.380s),0 (0.0V),681 (3.32V),1023 (5.0V)
430 (0.430s),0 (0.0V),681 (3.32V),1023 (5.0V)
480 (0.480s),0 (0.0V),680 (3.32V),1023 (5.0V)
520 (0.520s),0 (0.0V),681 (3.32V),1023 (5.0V)
570 (0.570s),0 (0.0V),681 (3.32V),1023 (5.0V)
620 (0.620s),0 (0.0V),681 (3.32V),1023 (5.0V)
670 (0.670s),0 (0.0V),681 (3.32V),1023 (5.0V)
720 (0.720s),0 (0.0V),681 (3.32V),1023 (5.0V)
770 (0.770s),0 (0.0V),681 (3.32V),1023 (5.0V)
820 (0.820s),0 (0.0V),681 (3.32V),1023 (5.0V)
870 (0.870s),0 (0.0V),681 (3.32V),1023 (5.0V)
910 (0.910s),0 (0.0V),680 (3.32V),1023 (5.0V)
960 (0.960s),0 (0.0V),681 (3.32V),1023 (5.0V)
1010 (1.10s),0 (0.0V),681 (3.32V),1023 (5.0V)
1060 (1.60s),0 (0.0V),681 (3.32V),1023 (5.0V)
1110 (1.110s),0 (0.0V),680 (3.32V),1023 (5.0V)
1160 (1.160s),0 (0.0V),681 (3.32V),1023 (5.0V)
1210 (1.210s),0 (0.0V),681 (3.32V),1023 (5.0V)
1260 (1.260s),0 (0.0V),681 (3.32V),1023 (5.0V)
1310 (1.310s),0 (0.0V),681 (3.32V),1023 (5.0V)
1360 (1.360s),0 (0.0V),681 (3.32V),1023 (5.0V)
1410 (1.410s),0 (0.0V),681 (3.32V),1023 (5.0V)
1460 (1.460s),0 (0.0V),680 (3.32V),1023 (5.0V)

COM14
Initializing SD card...card initialized.
40,0,681,1023
60,0,681,1023
80,0,681,1023
100,0,681,1023
120,0,681,1023
140,0,681,1023
160,0,681,1023
180,0,681,1023
200,0,681,1023
220,0,681,1023
240,0,681,1023
260,0,681,1023
280,0,681,1023
300,0,681,1023
320,0,681,1023
340,0,681,1023
360,0,681,1023
380,0,681,1023
400,0,681,1023
420,0,681,1023
440,0,681,1023
460,0,681,1023
480,0,680,1023
500,0,681,1023
520,0,681,1023
540,0,681,1023
560,0,681,1023
580,0,681,1023
600,0,681,1023
620,0,681,1023
640,0,681,1023
660,0,681,1023
```



Running the sketch

```
DATALOG.TXT - Notepad
File Edit Format View Help
126,0,682,1023
260,0,682,1023
288,0,682,1023
302,0,682,1023
314,0,681,1023
326,0,682,1023
338,0,682,1023
352,0,682,1023
364,0,682,1023
376,0,682,1023
390,0,682,1023
402,0,681,1023
414,0,682,1023
428,0,682,1023
440,0,682,1023
452,0,682,1023
464,0,682,1023
478,0,682,1023
490,0,682,1023
502,0,682,1023
516,0,681,1023
528,0,682,1023
540,0,682,1023
552,0,682,1023
566,0,681,1023
578,0,682,1023
590,0,682,1023
604,0,682,1023
616,0,682,1023
628,0,682,1023
642,0,682,1023
654,0,682,1023
666,0,682,1023
676,0,681,1023
690,0,682,1023
702,0,682,1023
714,0,682,1023
728,0,682,1023
740,0,682,1023
752,0,682,1023
764,0,682,1023
778,0,682,1023
790,0,682,1023
802,0,682,1023
816,0,682,1023
```

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 nowhere near the 2ms that we asked for.

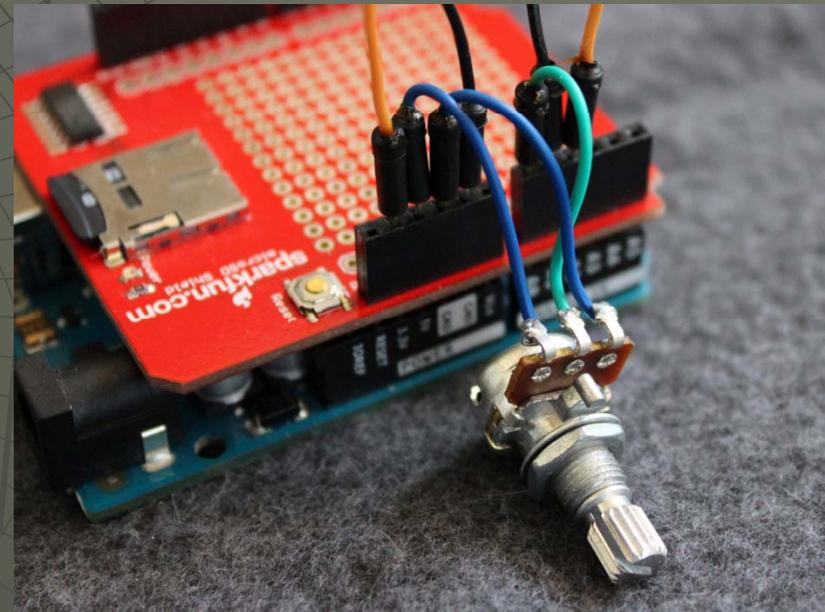
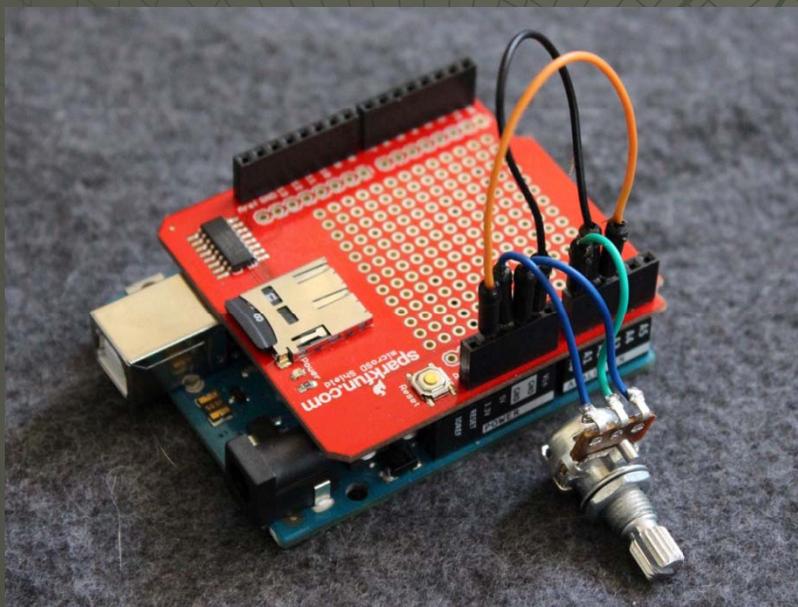
Be aware of times and what you are asking the Arduino to do when capturing data and be realistic

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

```
COM14
Initializing SD card...card initialized.
1000,0,0,681
2000,0,0,681
3000,70,0,681
4000,158,0,681
5000,233,0,681
6000,297,0,682
7000,358,0,681
8000,422,0,681
9000,471,0,681
10000,596,0,681
11000,696,0,681
12000,764,0,681
13000,856,0,681
14000,944,0,681
15000,1023,0,681
16000,1023,0,681
17000,932,0,681
18000,835,0,681
19000,757,0,681
20000,672,0,681
21000,603,0,681
22000,497,0,681
23000,414,0,681
24000,322,0,681
25000,221,0,682
26000,112,0,682
27000,0,0,681
28000,0,0,681
29000,0,0,681
30000,0,0,682
31000,0,0,681
32000,0,0,681
```

1

```
DATALOG.TXT - Notepad
File Edit Format View Help
1000,0,0,683
2000,0,0,682
3000,0,0,682
4000,0,0,682
5000,0,0,682
6000,0,0,682
7000,0,0,682
8000,0,0,682
1000,0,0,681
2000,0,0,681
3000,70,0,681
4000,158,0,681
5000,233,0,681
6000,297,0,682
7000,358,0,681
8000,422,0,681
9000,471,0,681
10000,596,0,681
11000,696,0,681
12000,764,0,681
13000,856,0,681
14000,944,0,681
15000,1023,0,681
16000,1023,0,681
17000,932,0,681
18000,835,0,681
19000,757,0,681
20000,672,0,681
21000,603,0,681
22000,497,0,681
23000,414,0,681
24000,322,0,681
25000,221,0,682
26000,112,0,682
27000,0,0,681
28000,0,0,681
29000,0,0,681
30000,0,0,682
31000,0,0,681
32000,0,0,681
33000,0,0,681
34000,0,0,681
```

2

```
DATALOG.TXT - Notepad
File Edit Format View Help
1000,0,0,683
2000,0,0,682
3000,0,0,682
4000,0,0,682
5000,0,0,682
6000,0,0,682
7000,0,0,682
8000,0,0,682
1000,0,0,681
2000,0,0,681
3000,70,0,681
4000,158,0,681
5000,233,0,681
6000,297,0,682
7000,358,0,681
8000,422,0,681
9000,471,0,681
10000,596,0,681
11000,696,0,681
12000,764,0,681
13000,856,0,681
14000,944,0,681
15000,1023,0,681
16000,1023,0,681
17000,932,0,681
18000,835,0,681
19000,757,0,681
20000,672,0,681
21000,603,0,681
22000,497,0,681
23000,414,0,681
24000,322,0,681
25000,221,0,682
26000,112,0,682
27000,0,0,681
28000,0,0,681
29000,0,0,681
30000,0,0,682
31000,0,0,681
32000,0,0,681
33000,0,0,681
34000,0,0,681
35000,0,0,681
```

3

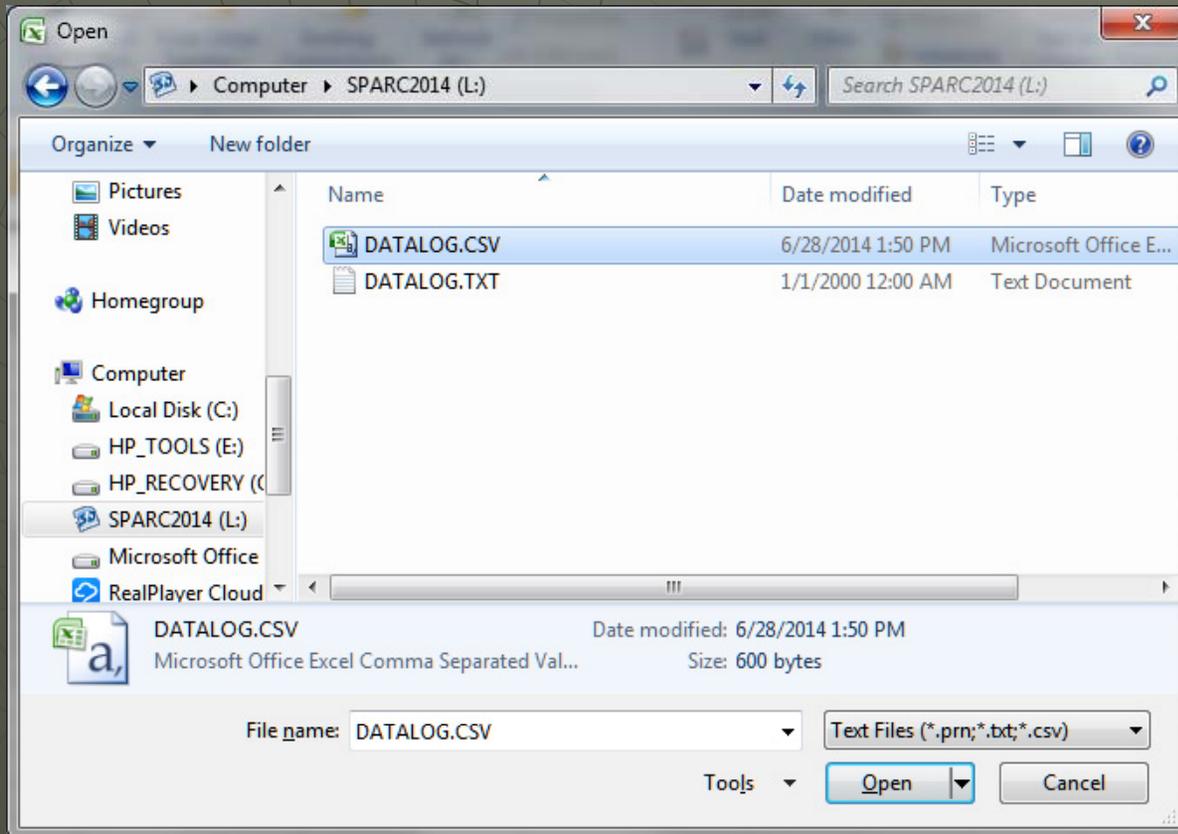
```
DATALOG.CSV - Notepad
File Edit Format View Help
1000,0,0,683
2000,0,0,682
3000,0,0,682
4000,0,0,682
5000,0,0,682
6000,0,0,682
7000,0,0,682
8000,0,0,682
1000,0,0,681
2000,0,0,681
3000,70,0,681
4000,158,0,681
5000,233,0,681
6000,297,0,682
7000,358,0,681
8000,422,0,681
9000,471,0,681
10000,596,0,681
11000,696,0,681
12000,764,0,681
13000,856,0,681
14000,944,0,681
15000,1023,0,681
16000,1023,0,681
17000,932,0,681
18000,835,0,681
19000,757,0,681
20000,672,0,681
21000,603,0,681
22000,497,0,681
23000,414,0,681
24000,322,0,681
25000,221,0,682
26000,112,0,682
27000,0,0,681
28000,0,0,681
29000,0,0,681
30000,0,0,682
```

4

- 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

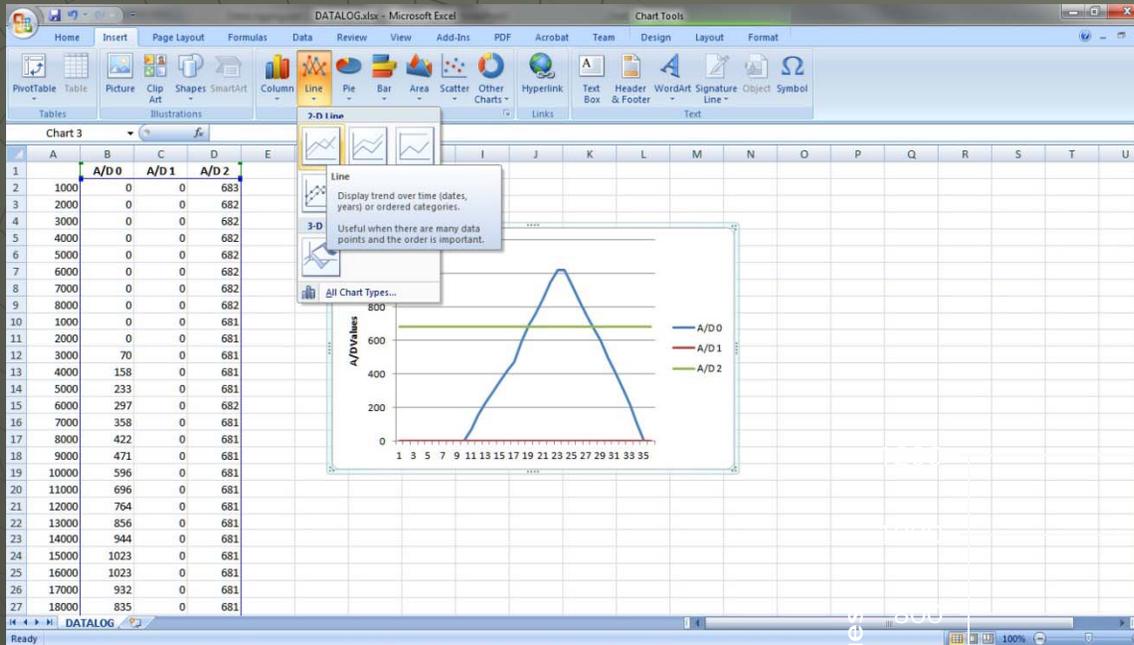


A screenshot of Microsoft Excel showing the data from the DATALOG.CSV file. The data is organized into columns A through G. Column A represents time, and columns B through G represent various A/D values.

	A	B	C	D	E	F	G
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			

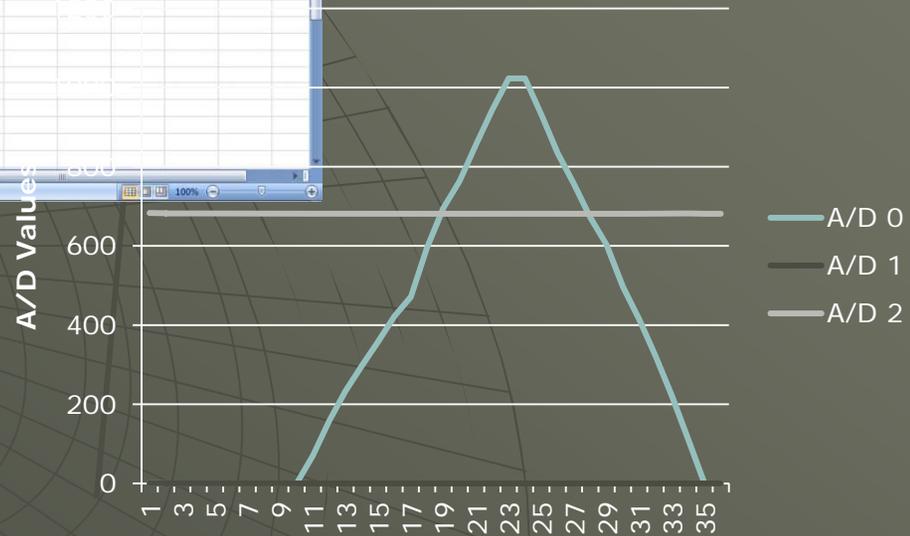
Start Microsoft Excel (or similar from Open Office) and open the DATALOG.CSV file – each value separated by a comma in the .csv file should be in its own cell in Excel – each row is a new time and each column should represent one of the A/Ds

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

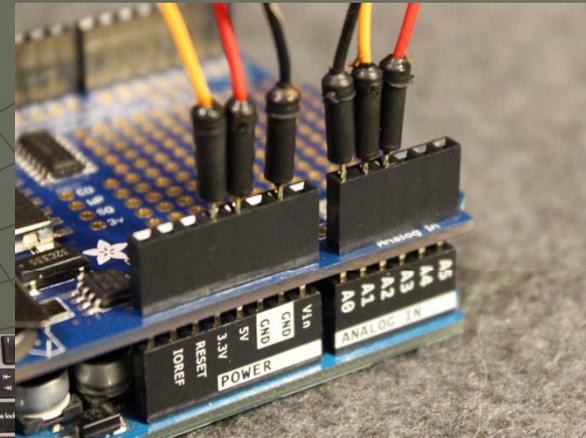
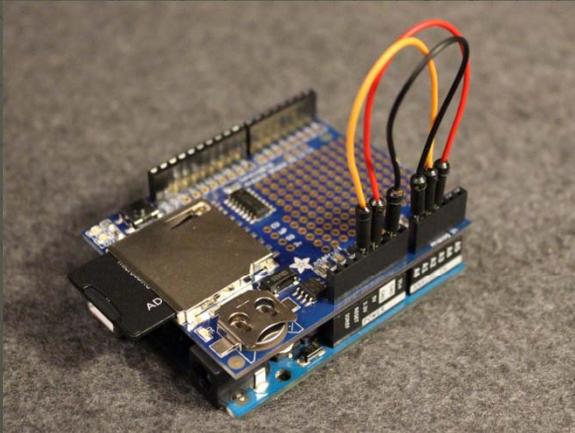
If you want to record the actual date and time instead of the time interval since the program started, you'll need a Real Time Calendar/Clock (RTC). The Adafruit shield provides the RTC as well as an interface for the SD Card (full sized – you'll need an adapter for the micro SD card





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



AIAA OC Rocketry

AIAA OC Section – NAR #718



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();  
}
```

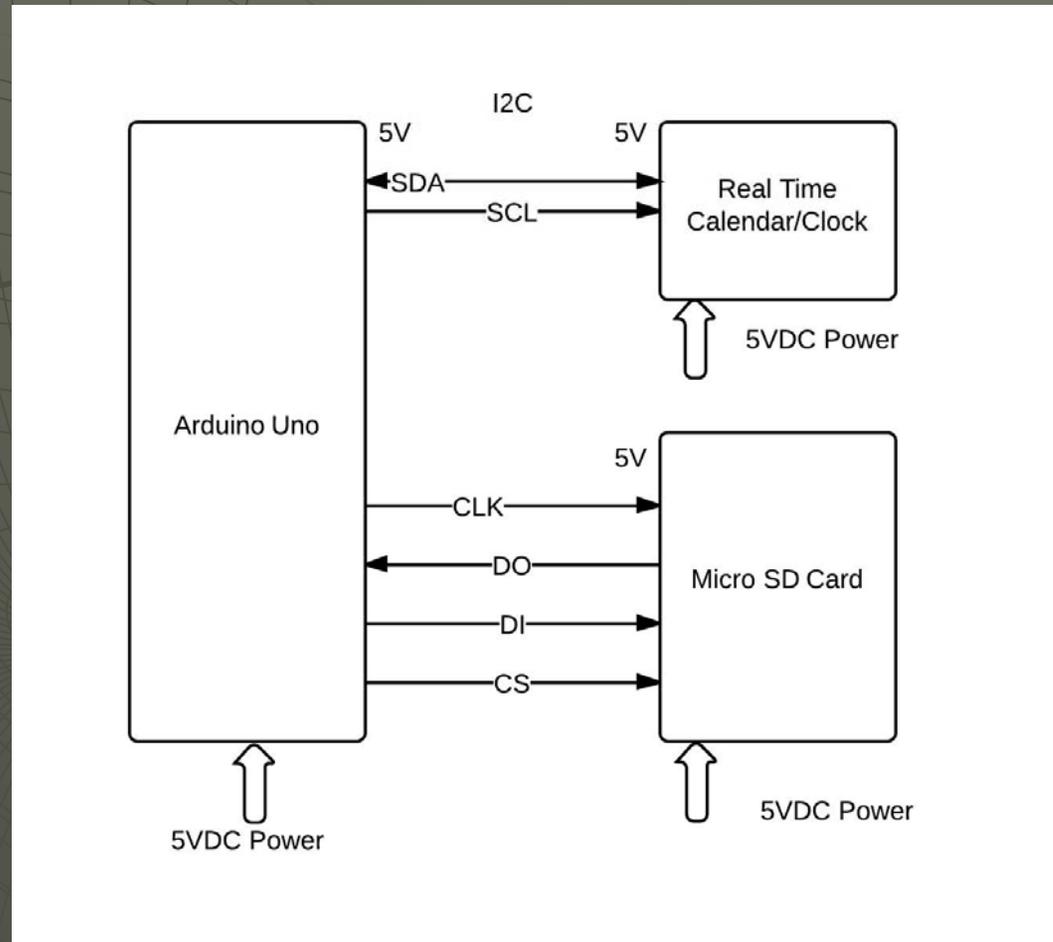
```
DATALOG.TXT - Notepad  
File Edit Format View Help  
Begin Data Capture: 2014/6/29  
5:25:0,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:1,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:2,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:3,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:4,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:5,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:6,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:7,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:8,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:9,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:10,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:11,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:12,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:13,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:14,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:15,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:16,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:17,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:18,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:19,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:20,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:21,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:22,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:23,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:24,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:25,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:26,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:27,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:28,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:29,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:30,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:31,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:32,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:33,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:34,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:35,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:36,0 (0.0v),681 (3.32v),1023 (5.0v)  
5:25:37,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:38,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:39,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:40,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:41,0 (0.0v),682 (3.33v),1023 (5.0v)  
5:25:42,0 (0.0v),682 (3.33v),1023 (5.0v)
```

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

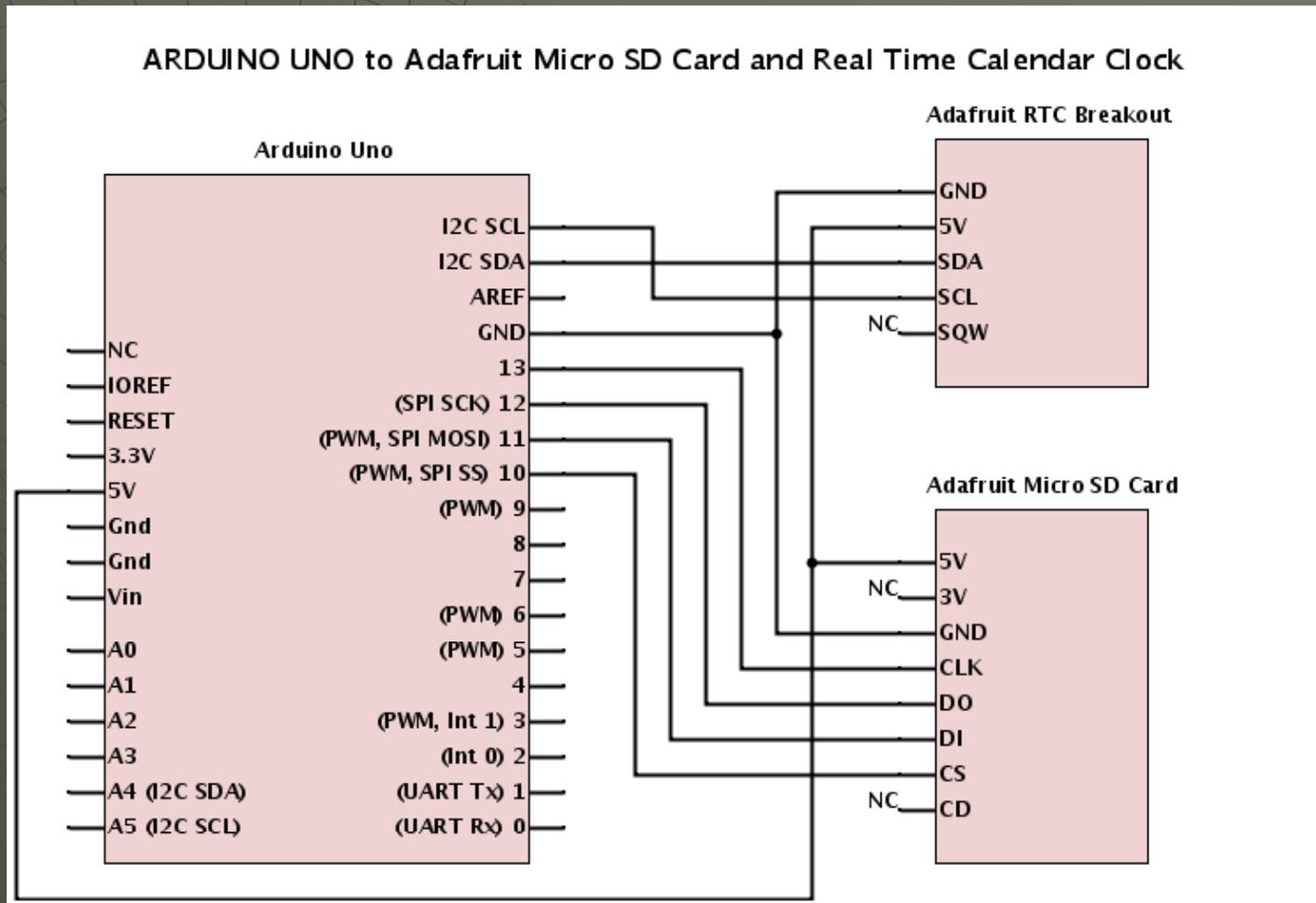


Connecting the Uno – Micro SD Card - RTC





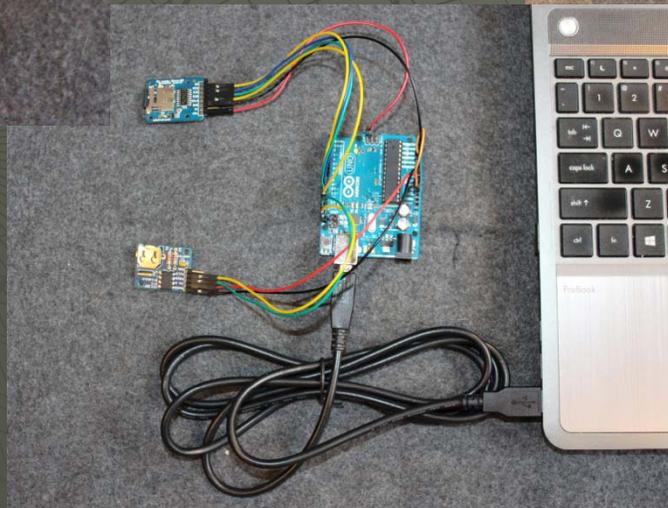
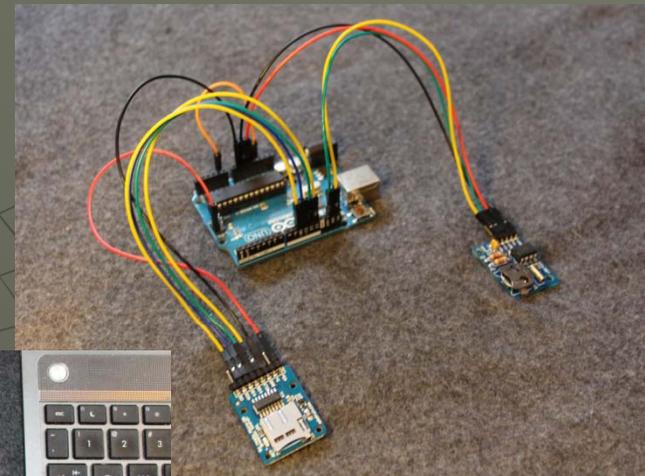
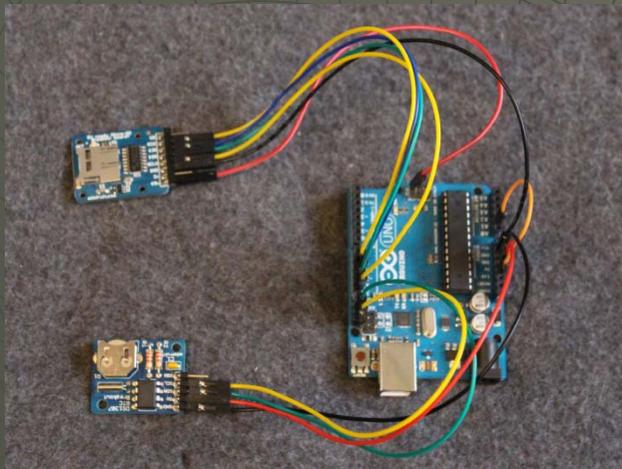
Connecting the Uno – Micro SD Card - RTC





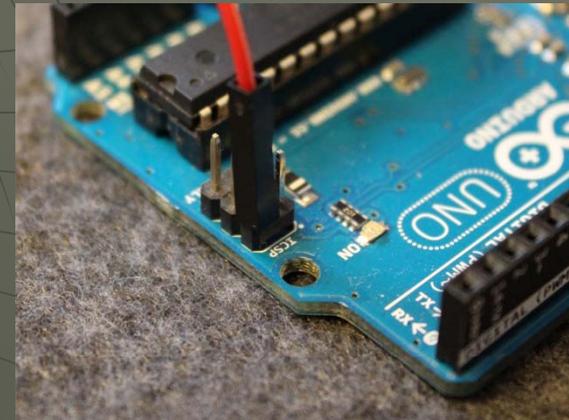
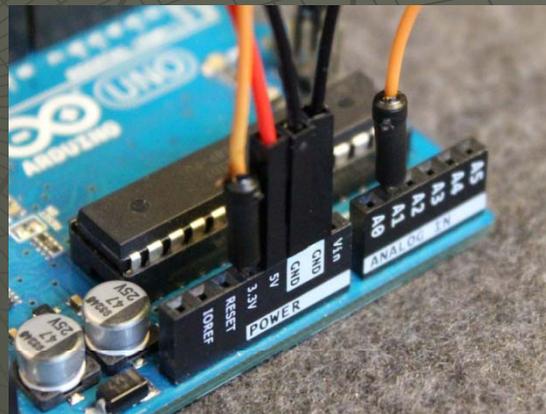
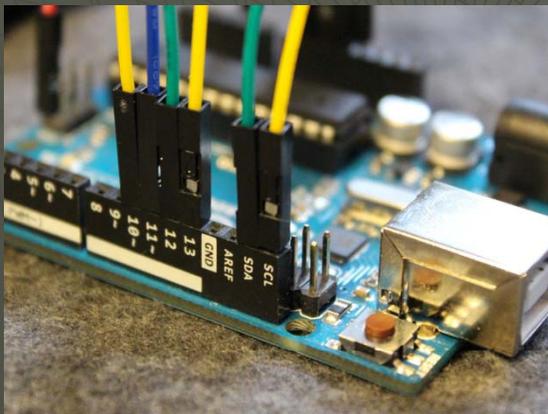
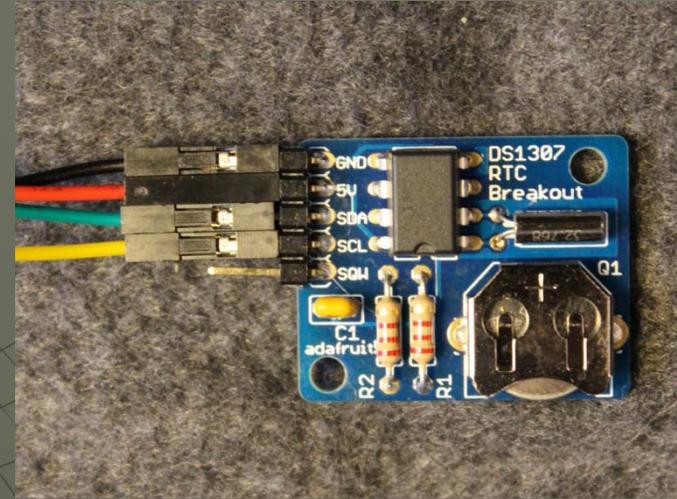
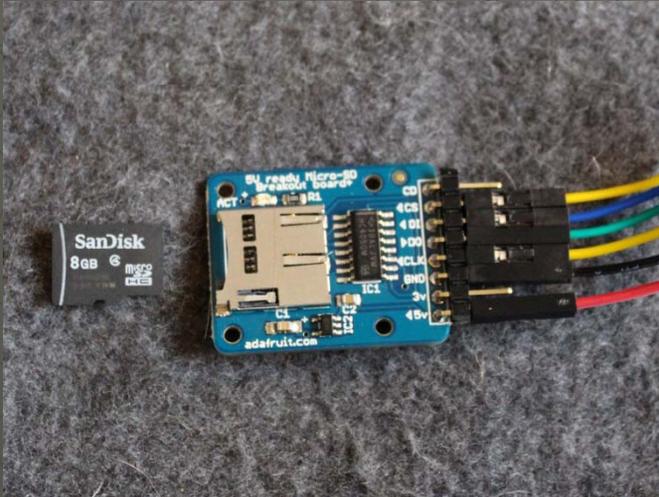
Setting up the discrete hardware

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



AIAA OC Rocketry

AIAA OC Section – NAR #718

For the RTC breakout board hardware we'll use the same SD_Datlogger._with_RTC with no changes. If this is 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.

```
COM14
Initializing SD card...card initialized.

Begin Data Capture: 2014/6/29
8:7:44,389 (1.90V),679 (3.31V),538 (2.62V)
8:7:45,375 (1.83V),680 (3.32V),590 (2.88V)
8:7:46,403 (1.96V),680 (3.32V),609 (2.97V)
8:7:47,415 (2.2V),680 (3.32V),617 (3.1V)
8:7:48,420 (2.5V),680 (3.32V),620 (3.3V)
8:7:49,421 (2.5V),681 (3.32V),622 (3.4V)
8:7:50,423 (2.6V),681 (3.32V),622 (3.4V)
8:7:51,423 (2.6V),681 (3.32V),623 (3.4V)
8:7:52,424 (2.7V),682 (3.33V),623 (3.4V)
8:7:53,424 (2.7V),681 (3.32V),623 (3.4V)
8:7:54,425 (2.7V),681 (3.32V),623 (3.4V)
8:7:55,425 (2.7V),682 (3.33V),623 (3.4V)
8:7:56,425 (2.7V),681 (3.32V),623 (3.4V)
8:7:57,425 (2.7V),681 (3.32V),623 (3.4V)
8:7:58,426 (2.8V),681 (3.32V),623 (3.4V)
8:7:59,426 (2.8V),681 (3.32V),623 (3.4V)
8:8:0,426 (2.8V),681 (3.32V),623 (3.4V)
8:8:1,427 (2.8V),681 (3.32V),623 (3.4V)
8:8:2,427 (2.8V),681 (3.32V),623 (3.4V)
8:8:3,427 (2.8V),681 (3.32V),622 (3.4V)
8:8:4,427 (2.8V),681 (3.32V),622 (3.4V)
8:8:5,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:6,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:7,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:8,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:9,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:10,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:11,428 (2.9V),681 (3.32V),621 (3.3V)
8:8:12,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:13,425 (2.7V),681 (3.32V),622 (3.4V)
8:8:14,426 (2.8V),682 (3.33V),622 (3.4V)
```

```
DATALOG.TXT - Notepad
File Edit Format View Help

Begin Data Capture: 2014/6/29
8:7:44,389 (1.90V),679 (3.31V),538 (2.62V)
8:7:45,375 (1.83V),680 (3.32V),590 (2.88V)
8:7:46,403 (1.96V),680 (3.32V),609 (2.97V)
8:7:47,415 (2.2V),680 (3.32V),617 (3.1V)
8:7:48,420 (2.5V),680 (3.32V),620 (3.3V)
8:7:49,421 (2.5V),681 (3.32V),622 (3.4V)
8:7:50,423 (2.6V),681 (3.32V),622 (3.4V)
8:7:51,423 (2.6V),681 (3.32V),623 (3.4V)
8:7:52,424 (2.7V),682 (3.33V),623 (3.4V)
8:7:53,424 (2.7V),681 (3.32V),623 (3.4V)
8:7:54,425 (2.7V),681 (3.32V),623 (3.4V)
8:7:55,425 (2.7V),682 (3.33V),623 (3.4V)
8:7:56,425 (2.7V),681 (3.32V),623 (3.4V)
8:7:57,425 (2.7V),681 (3.32V),623 (3.4V)
8:7:58,426 (2.8V),681 (3.32V),623 (3.4V)
8:7:59,426 (2.8V),681 (3.32V),623 (3.4V)
8:8:0,426 (2.8V),681 (3.32V),623 (3.4V)
8:8:1,427 (2.8V),681 (3.32V),623 (3.4V)
8:8:2,427 (2.8V),681 (3.32V),623 (3.4V)
8:8:3,427 (2.8V),681 (3.32V),622 (3.4V)
8:8:4,427 (2.8V),681 (3.32V),622 (3.4V)
8:8:5,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:6,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:7,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:8,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:9,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:10,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:11,428 (2.9V),681 (3.32V),621 (3.3V)
8:8:12,428 (2.9V),681 (3.32V),622 (3.4V)
8:8:13,425 (2.7V),681 (3.32V),622 (3.4V)
8:8:14,426 (2.8V),682 (3.33V),622 (3.4V)
8:8:15,428 (2.9V),681 (3.32V),621 (3.3V)
8:8:16,427 (2.8V),681 (3.32V),621 (3.3V)
8:8:17,427 (2.8V),682 (3.33V),621 (3.3V)
8:8:18,426 (2.8V),681 (3.32V),621 (3.3V)
8:8:19,426 (2.8V),681 (3.32V),621 (3.3V)
8:8:20,426 (2.8V),681 (3.32V),621 (3.3V)
8:8:21,427 (2.8V),681 (3.32V),621 (3.3V)
8:8:22,427 (2.8V),682 (3.33V),621 (3.3V)
8:8:23,427 (2.8V),681 (3.32V),622 (3.4V)
8:8:24,428 (2.9V),681 (3.32V),621 (3.3V)
8:8:25,428 (2.9V),681 (3.32V),621 (3.3V)
8:8:26,429 (2.9V),682 (3.33V),621 (3.3V)
```

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>