Statement of Purpose

The purpose of this project was to build a system to measure the voltage waveform of a discharging railgun. Another group of students had built a railgun system to simulate micro asteroid impacts in low Earth orbit. The only instrumentation available to them was an analog voltmeter attached to the capacitor banks. During the lengthy (30 to 45 minute) charging process, an analog voltmeter was more than capable of measuring the voltage. However, during the discharging process the time elapsed was so short (approximately one millisecond) that waveform data could not be captured. In order to measure power dissipation during discharge as well as the energy put into the projectile, a system to measure the waveform of the discharging state needed to exist.

System Requirements

The system requirements were specified by the team building and testing the railgun, the system built had to meet the following specifications:

- Perform at least 1000 measurements in the millisecond discharge
- Be able to save at least 7 milliseconds worth of data (3 milliseconds prior to firing, 1 millisecond during firing, 3 milliseconds after firing)
- Have at least 1V accuracy
- Accept $V_{in}$ from 0V to 500V
- Require as few inputs as possible to allow for the system to be shielded against magnetic fields generated when railgun fires
- System needs to be less than $200
- Measurement system should not adversely affect railgun system

Hardware Specifications

- mbed 1768 Microcontroller

The small size of the mbed 1768 coupled with the low price and high clock speed made the microcontroller choice fairly simple. Two issues came up with the use of the mbed 1768. The first is that it is required to use the mbed online compiler, which can be useful for team projects, but does mean that in order to update the microcontroller an internet connection is required. The other issue is that the mbed 1768 is listed as having 64K of onboard RAM, but this is incorrect. Half of that is reserved for peripherals, giving only 32K of available memory. This is enough to meet the requirements set forth by the railgun team, but required some design changes.

- ADCS 7477 10-bit ADC

For at least 1V accuracy across a 500V wave the ADC would need at least 9 bits ($500V / 2^{9}$
levels = 0.977 V/level). To satisfy the 1000 samples per millisecond (spms) requirement the ADC would need to be able to capture 1 million samples per second. This requirement immediately made using the onboard ADCs of a microcontroller not an option, because the fastest that the mbed 1768 ADC’s will operate at is 200ksps, only 20% of the required capture rate. The ADCS 7477 met the necessary requirements, but did require the additional purchase of a SOT-23-6 to DIP breakout board. The minimum clock requirement for the ADCS 7477 to operate at 1Mmps is 20Mhz, well within what the mbed 1768 can put out.

- SD Card Breakout Board

Due to the shielding requirement, the measurement system needed to be as self-contained as possible. This meant that data would have to be stored locally, and then expect to be used later by the railgun team. The ease of use, low cost, and breakout board availability of SD cards made SD cards the best choice for storage.

- Voltage Divider

In order to successfully integrate with the railgun a 100:1 voltage divider needs to be created to get the $V_{\text{max}}$ from 500V to 5V, where it is actually measurable by the ADC that has been purchased. In order to prevent any potential issues with parasitic capacitance a capacitive divider in parallel with the resistive divider is used (see: Figure 1).

![Figure 1: Voltage divider setup.](image)

In order for the measurement system to impact the railgun system as little as possible, the power draw of the measurement system should be very minor. At $V_{\text{max}}$, the difference between $V_{\text{DD}}$ and $V_{\text{SS}}$ is 500V, and the measurement system should be using 0.5 W of power, maximum.
\[ P = V^2/R \]
\[ 0.5W = (500 V)^2/R \]
\[ R = 250000 \frac{V^2}{0.5W} \]
\[ R = 500k\Omega \]

The maximum input current allowed into the ADC is 10mA, which is well within the expected current values. Even with the smallest allowable resistor values, the most current being drawn would not be higher than 1mA. With this minimum value established an \( R_{\text{large}} \) of 511kΩ was selected, with a paired \( R_{\text{small}} \) of 5.11kΩ. In order to choose the capacitors the most important statistic was the voltage rating of the \( C_{\text{small}} \) capacitor. Any capacitor with a voltage rating greater than 500V and a large enough capacitance to nullify parasitic capacitance was acceptable.

### System Architecture

As shown in Figure 2, the software architecture of the system is effectively split into three sections. During the first section the system is operating in a low-power state. This is set by lowering the clock rate for the ADC SPI to limit the power drain from unnecessary measurements. Due to the lengthy charging time of the railgun capacitor banks, it is a waste of power to constantly measure at 1Msps. Instead, measuring at the minimum clock speed of 20 kHz until a ‘threshold’ voltage is reached is preferable. This means that measuring at the maximum speed will only occur for the final few minutes of charging and firing instead of the entire 30-45 minute charging period. The threshold is defined in the code of the microcontroller, and can be altered for shorter or longer charging times by either decreasing or increasing the threshold value respectively.
Once the threshold value has been passed we start filling the ‘pre-trigger’ list. This is a list of values that are captured prior to the discharge of the railgun. As shown in Figure 2, this ‘pre-trigger’ list needs to be constantly updated with a pointer to the oldest recorded value. This allows the program to know which value in the array should be replaced the next time that a value is captured, or once we start writing the data, remembers where to start writing data from.
while (val < THRESHOLD) {
    adccs = 0;
    val = adcspi.write(0x00);
    adccs = 1;
}

adcspi.frequency(FASTCLK);

while (oldval - ACCVAL <= val) {
    prearray[count] = val;
    oldval = val;
    count++;
    if (count == ARRAYSIZE) {
        count = 0;
    }
    adccs = 0;
    val = adcspi.write(0x00);
    adccs = 1;
}

Figure 3: The pre-trigger code that updates the pre-array.

As can be seen in Figure 3, the filling of the pre-trigger array will continue until a drop in voltage is measured. This drop is altered by the ACCVAL, a defined value that will be modified based on the accuracy of the ADC. This value is calculated based on how accurate the ADC is measured to be so that noise in the ADC capture doesn’t get translated to the railgun firing.

One thing that should be noted is that due to the mbed 1768 library, it is assumed that all SPI peripherals will have both a MOSI and MISO connection. This is not the case for the ADCS 7477, as it only contains a MISO connection. The only SPI call available is write(), which both writes to the SPI and reads the response. In this case, a dummy write is called so that the read can be performed, but what value is being written doesn’t matter.

Once the ‘trigger’ is tripped, the post-trigger array starts to fill with captured values. Once the post-trigger array is filled, the data captured is written to the SD card. Due to the capture rate (1Mps) a measurement is captured every microsecond. This allows the measurement number to also be used as the time value on the x-axis. A sample capture and output performed by the onboard ADC can be seen in Figure 3. Note that the graph is slightly skewed due to the saw wave input having a different timing than what would be expected in the real world (only a 0.5 second charging time instead of 30-45 minutes). The upside is that this allows confirmation that the threshold measurement is working properly, as well as the output of the values in both the pre-trigger and post-trigger arrays.
The complete wiring diagram can be seen in Figure 4, note that the SD Card breakout board does not use every pin. Due to using the card as SPI, it is wired according to Figure 5.
The wired system can be seen below in Figure 6. Note that the 9V battery connection, switch, and 5V regulator are not yet part of the system because the 9V snap connector and switch are two of the parts that were lost in shipping.
Figure 5: Assembled system, from the top down there is the microcontroller, SD breakout board, ADC, and voltage divider. The yellow cable to the right would be the input and the orange cable to the left would be ground.

System Tests and Problems Encountered

The system was tested as separate pieces at first. To start with, the ideal voltage divider values were plugged into PSpice to simulate the perfect result (Figure 6). Once those values were satisfactory and the parts were delivered, the measured values were plugged into the same test circuit to get an expected result (Figure 7). This expected result was close enough to the perfect that there wasn’t a need to add further components, although to get a voltage divider closer to 100:1 accuracy adding additional resistors or capacitors is an option.
Figures 6 and 7: (Left) The simulated results with perfect components. (Right) Simulated results with measured values of delivered components.

In order to prove the voltage divider as accurate it was tested with a 10V sawtooth input in order to simulate the expected long charging time and short discharging time that would be experienced in the real-world environment. Due to the 100:1 voltage division the resulting maximum output is only 100mV. In order to get a proper output waveform measurement a sync cable needs to be connected between the oscilloscope and waveform generator. Once this was done I was able to confirm that the voltage divider matched the simulated values produced in PSpice.

After the voltage divider test the other components were tested starting with the SD card breakout board. A simple “Hello, world!” test can be found on the mbed SDCardLibrary page and showed that the connections between the mbed microcontroller and the SD card breakout board was working properly. To help with debugging in the future, serial communication between the microcontroller and attached computer was placed into the program. Once this was complete, I continued testing by hooking up the ADCS 7477. Due to a wiring error (note that the 3.3V output and the 5V output are directly next to each other on the microcontroller) the ADCS 7477 was seriously damaged. With the offboard ADC damaged the onboard ADC had to be used to test the written code.
Due to the use of the onboard ADC note that the maximum value of the capture is 65536 \((2^{16})\) instead of 4096, like would be expected from the ADCS 7477.

The other major issue that came up was that the railgun system itself was damaged and not repaired in time for testing to be done on that system. So it is an unknown as to how well this system works in the railgun environment.

**Future Integration**

Due to the shipping error some of the parts (replacement ADC, switch, 9V snap connector) need to be reordered and added into the system. Both sets of code (onboard and offboard ADC support) will be provided to help with the ADC integration. One note is that the onboard ADCs of the mbed board have a \(V_{\text{max}}\) of 3.3V, so if comparison testing is performed between the onboard and offboard ADC make sure to not exceed that. Software implementation notes and comments are inside the code, the major values that will need to be changed are the ACCVAL and THRESHOLD values, depending on the accuracy and target threshold respectively.

**User Manual**

Note that this user manual is assuming operation with the ADCS 7477 as well as the other parts that need to be replaced due to the shipping error.

1. Connect input and ground lines
2. If you would like serial communication with a computer for testing:
   a. Disconnect 9V battery
b. Connect microcontroller to computer using USB-microUSB cable
c. Run your preferred serial connection program (I used PuTTY) at 9600 baud
d. Skip step 3

3. Turn on switch
4. Wait for discharge
5. After discharge is complete, remove SD card
6. Plug SD card into computer, the reading will be in “/sd/mydir/sdtest.csv”
7. Open .csv file in Excel or similar program
Appendices

Datasheets:

ADCS 7477 Datasheet:

LPC1768 Microcontroller:

SD Card Writer
http://dlhmh9ip6v2uc.cloudfront.net/datasheets/Components/General/SDMMCCardSock.pdf

5V Regulator
http://www.micrel.com/_PDF/mic2920.pdf

Code:

Code for use with ADCS 7477:
http://mbed.org/users/stumpster/code/HVWaveformSensor/

Code using the mbed 1768 onboard ADC:
http://mbed.org/users/stumpster/code/HVWaveformSensorOB/

Bill of Materials:

<table>
<thead>
<tr>
<th>Piece</th>
<th>Price</th>
<th>Part #</th>
<th>Ordered From</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-bit ADC</td>
<td>3.27</td>
<td>ADCS 7477</td>
<td>Digikey</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>59.95</td>
<td>mbed 1768</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>SD Breakout Board</td>
<td>9.95</td>
<td>BOB-11403</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>5V Regulator</td>
<td>3.42</td>
<td>MIC2920A-5.0WT</td>
<td>Digikey</td>
</tr>
<tr>
<td>511k Resistor</td>
<td>0.77</td>
<td></td>
<td>Digikey</td>
</tr>
<tr>
<td>5.11k Resistor</td>
<td>1.92</td>
<td></td>
<td>Digikey</td>
</tr>
<tr>
<td>68pF Capacitor</td>
<td>0.34</td>
<td></td>
<td>Digikey</td>
</tr>
<tr>
<td>6800pF Capacitor</td>
<td>1.01</td>
<td></td>
<td>Digikey</td>
</tr>
<tr>
<td>SOT-23-6 to DIP</td>
<td>0.95</td>
<td>BOB-00717</td>
<td>Sparkfun</td>
</tr>
<tr>
<td>Item</td>
<td>Price</td>
<td>Supplier</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>SD Card</td>
<td>5.93</td>
<td>Amazon</td>
<td></td>
</tr>
<tr>
<td>Breadboard + Wiring</td>
<td>13.82</td>
<td>Digikey</td>
<td></td>
</tr>
<tr>
<td>9V Snap Connector</td>
<td>0.59</td>
<td>Digikey</td>
<td></td>
</tr>
</tbody>
</table>