Project 3
Building a Data Acquisition and Storage Board for a Strain Gauge

Peregrine Badger, Tim McNamara, Liam Mulshine, Sandra Schumann

May 9, 2016

ES91r: The Frustration of Electronics

1. Introduction

In this project we set out to build a data acquisition system for the above load sensor. The original design brief suggested benchmarks for data input rate, accuracy, number of stored samples, and user interface, and left the specific use case open ended. Thus, we made design decisions that sought to balance difficulty and simplicity, so that we would be challenged during the design process while also having something that worked at the end. Thus, our specs are set more based on their perceived complexity than a real-world application. Learning goals surrounding this project were focused on noise reduction and getting a clear signal from the sensor, which required an understanding of ground loops, noise ratios, and common vs. differential modes.

2. Design Process

2.1 Problem and Project Specifications

As with any design project, we began by defining the problem we were looking to solve, and constructing a set of specifications that would meet that solution. The problem was mostly defined for us: we needed to interface with a specific load cell, capture useful information about the forces being applied to it, interpret this data, and save it onto an SD card. Our initial specifications were defined as follows:

- Able to capture events with frequencies as high as 1kHz
- f_s > 60 kHz (Oversampling by factor of ~30)
    - Sampling frequency set by microcontroller

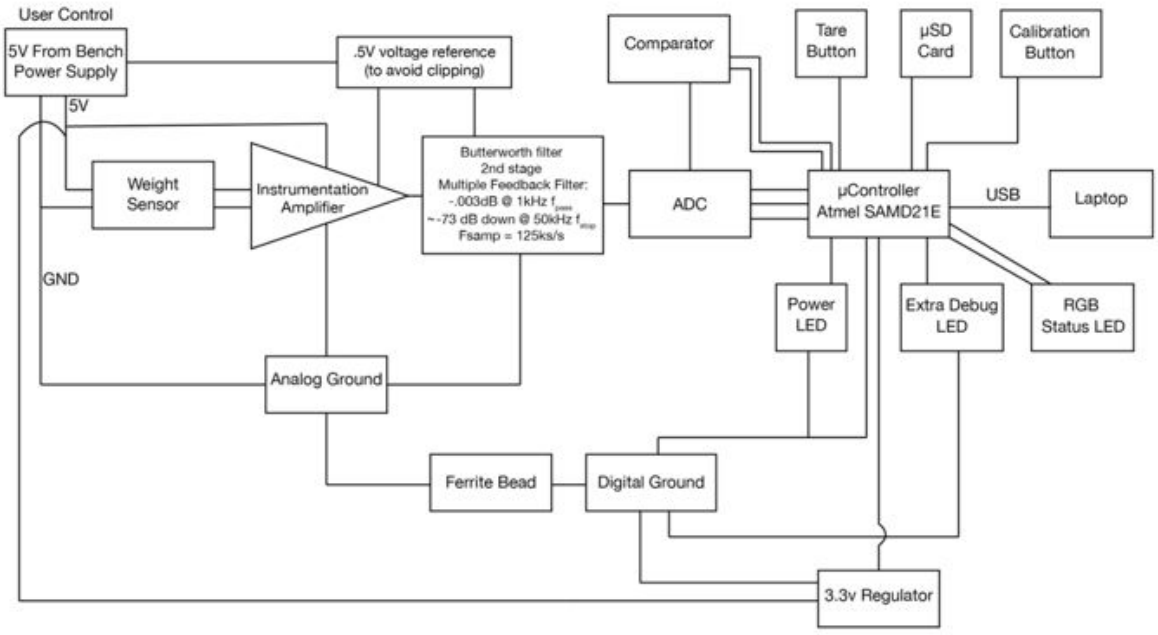
- Accuracy within 1 lbf (1% FSV)
- 256 Mbytes SD storage
- Powered with 5V from Lab Power Supply
- USB interface between uC and PC for data transfer from uSD card to PC
- User interface consisting of pushbuttons inputs and LED feedback
- Cost < $50

Soon after putting together the firmware for the M0 microcontroller, we realized that sampling at such a high frequency using the timer-counter peripheral would be difficult given the limitation of the microcontroller. Therefore, we were forced to reduce our specification to an achievable sampling frequency of ~2940Hz. This sampling frequency exceeds the Nyquist frequency for a 1kHz signal, so our goal to capture 1kHz events was within reach.

With our problem and our specifications well defined, we began to design our circuit.

2.2 Block Diagram

The block diagram for our circuit is provided below.



A major focus of this project was precision. Therefore, considerable emphasis was placed on removing sources of noise from our design and, if that proved impossible, reducing their effect on the signal as much as possible. The goal was to get the signal from the load cell to the digital

realm as quickly as possible, since, once in digital, the signal is essentially unaffected by unpredictable analog noise, and rarely loses integrity.

One important step we took in reducing noise levels along our signal path was separating the analog and digital grounds, and placing all digital components as far from the analog components as possible. Please refer to the Layout section for a detailed description of the considerations made in this regard.

The load-cell provides a differential signal, helping to increase signal integrity, since, assuming an appropriate layout, any stray inductance in the cable would affect both signals equally. However, we still recognized the need to pass the signal through an antialiasing filter before analog to digital conversion to avoid aliasing. Thus, we decided to make the incoming signal single-ended to reduce circuit complexity, and prevent the introduction of unnecessary noise to the circuit. We amplified the signal using an instrumentation amplifier, which boasts a high common mode rejection ratio, and low dependence on external component intolerances. Since sending a weak signal through a filter would likely increase the signal to noise ratio to unacceptable levels, we adjusted the gain of our instrumentation amplifier so that the outcoming signal spanned the entire extent of the ADC's dynamic range. After adding gain, the signal to noise ratio was reduced considerably; therefore, the effects of noise on our signal were significantly reduced. As can be seen in our block diagram, our filter specs were originally quite high. In the end we needed to reduce these specifications, due to the limitations in setting a high sampling frequency with the M0 microcontroller.

After adding gain to and filtering the single-ended signal, our signal was properly conditioned to pass into the digital realm through the analog-to-digital converter. The converter's sampling frequency is set by the microcontroller. Please refer to Firmware section of this document for a detailed description on how this is done.

From there, we were predominantly dealing with digital signals. One unique feature of our circuit is event detection. We include a comparator that interfaces with the microcontroller, and toggles a digital signal when it senses an "event" at the load cell. This allows the microcontroller to be interrupt driven, entering sleep mode when no load is present. Our circuit also includes a simple user interface consisting of various pushbuttons and an RGB LED for user feedback.

2.3 Part Selection

Instrumentation Amplifier:

Seeing it as unnecessary to build our own instrumentation amplifier with discrete parts, we decided on the LTC1789, a low power, single-supply rail-to-rail instrumentation amplifier with gain set by a single external resistor. Given that we wanted a gain of 33, we chose the gain resistor to be 6.25kΩ, per instructions on the datasheet. It also provided relatively high PSRR

and CMRR (~100dB) and low voltage noise, characteristics which were crucial to maintain signal integrity in our application.

Anti-aliasing filter:

When designing the anti-aliasing filter, we considered the desired sampling rate, our capabilities of oversampling and desired accuracy. We originally decided to use a filter that would give us attenuation less than -0.08dB (less than 1 part in 100) at 1kHz. For our stop band, we looked at the maximum sampling rate of the ADC (250kHz) and decided to sample at half of that (125kHz). This would allow us to sample at a rate the ADC is capable of doing and also have a fair amount of attenuation at frequencies two times smaller than the sampling frequency. More precisely we decided to aim for attenuation more than -40dB (more than 99 parts in 100) at 50kHz (less than half of 125kHz). The reasoning for setting the sampling frequency so high was that we were aiming for having a relatively low-order filter in order to have a small number of stages and components. The less components, the less noise and less chance of the design not working.

We entered the aforementioned constraints into Texas Instruments filter design tool (see References), which we used to design a filter. This gave us a number of different filter options. We decided to go with a 2nd order Butterworth filter because of its simplicity and relatively low number of components. It also satisfies the constraints listed above and we didn't have to worry about the size of the passband ripple (unlike in for example Chebyshev filter). Of the specific topologies, we discovered using TI's helpful simulation tool that while the Sallen-Key topology seems to give us sufficient attenuation at 50kHz, the attenuation according to the simulator was less at some higher frequencies. At the same time the multiple feedback topology did not have the same problem, so we decided to go with that.

We also calculated the size of buffer that we would need in order to record data at rate 125ksps for 5 seconds. If every data point is stored in a 4-byte integer, storing all the data from 5 seconds in memory would take about 2.5MB of RAM. While we did know that our microcontroller only has 32kB of SRAM, for some reason this mismatch did not cross our minds. However, due to us later sampling at a much lower rate than originally intended, this ended up not being a problem (see the Results section).

For the op-amp in the filter, we chose the LMC6482, a part we had all worked with and one we knew could operate rail-to-rail, which was desirable for our single-supply design. However, we managed to make a mistake in our single-supply design after all (see Testing and Debugging).

Analog-Digital Converter:

To achieve higher bit resolution, we chose not to use the ADC included in the microcontroller, and settled on the LTC1864, a 16-bit ADC with SPI interface. This 8-pin part appeared not only

to be quite accurate, and allow sampling rates as high as 250 ksps, but also quite straightforward to interface with.

Level Shifter:

Since we needed to establish communication between the microcontroller and the analog to digital converter, we needed a way to shift logic levels up from 3.3V to 5V when sending information to the ADC, and down from 5V to 3.3V when sending information to the microcontroller. Avi suggested that we use a level shifter for this purpose. Since we needed two-way communication, we set out to find a bi-directional level shifter. We thought that we had found one in the MAX3391EEUD+, but it turned out that the '+' version was unidirectional. What we really needed was the original MAX2291EEUD. As explained in the debugging section, we quickly determined that we had ordered the unidirectional part, and had to find a bidirectional level shifter that could fill in. Luckily, we had the SparkFun Bi-Directional Logic Level Converter in lab that, while making our pcb look like a millipede, did the job well.

SAMD21e:

The main reason that we decided to use the SAMD21E was that we had already had experience working with the part. We also did not believe that it would significantly limit us in any particular way, despite the fact that it is an M0 microcontroller, which is less robust than most other ARM microcontrollers (such as M3s and up), with respect to CPU speed, ability to handle and store large amounts of data, etc. We later realized that the M0 did not allow us to achieve quite as high of a sampling frequency as we had initially desired; however, it still allowed us to capture as high as 1000 Hz events without risking aliasing.

3.3V LDO:

In order to power the SAMD21E we needed a 3.3V rail. We decided to use the AS1363 to generate this, since we had worked and had luck with the part in previous projects. This low power linear regulator is very stable, and has a very small dropout voltage, which would be beneficial if we someday decided to use batteries to power our circuit.

Comparator - LM311:

The LM311 was the classic comparator that we used in ES52. The reason that we decided to use this part initially was that it allowed us to distinguish between the supply voltages and the logic level voltages. Thus, we could power the comparator off of the 5V rail, but have it output a logic HIGH of 3.3V. Ultimately, we ended up using the microcontroller's internal input pull-up functionality, however, still stuck with the LM311, because of the familiarity we had with the part.

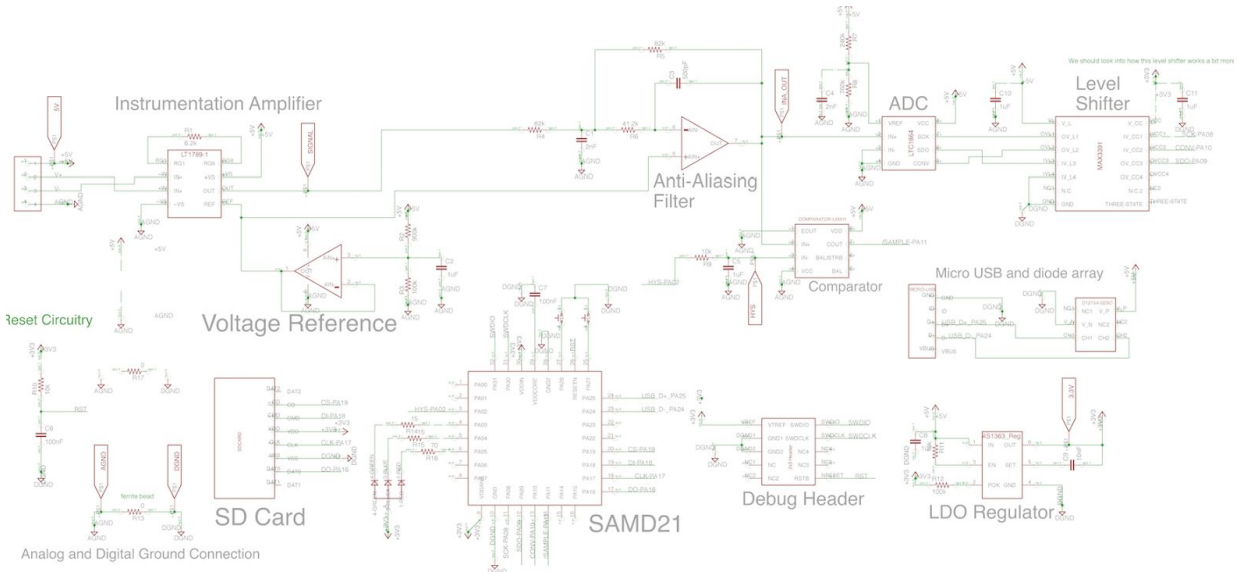
Generic USB-B:

Initially we designed with micro-USB connector to provide a point of connection between our board and a PC. However, after Avi pointed out how difficult such component can be to solder, and after considering that there were few size constraints in the project, we decided to use a USB-B connector instead. This connector, while considerably larger than the micro connector, is much easier to interface with. For a first revision, we decided, we would rather have a reliable connection and ease the pains of debugging, than have our board look pretty. For the µSD port, we found a standard one on-line.

RGB LED:

The RGB LED that we decided to use in this project has been another fall-back part this semester. The benefit of the RGB LED is that it can give you more than just binary (on-off) feedback. Having one LED that can turn different colors on command has proved considerably helpful in debugging.

2.4 Schematic



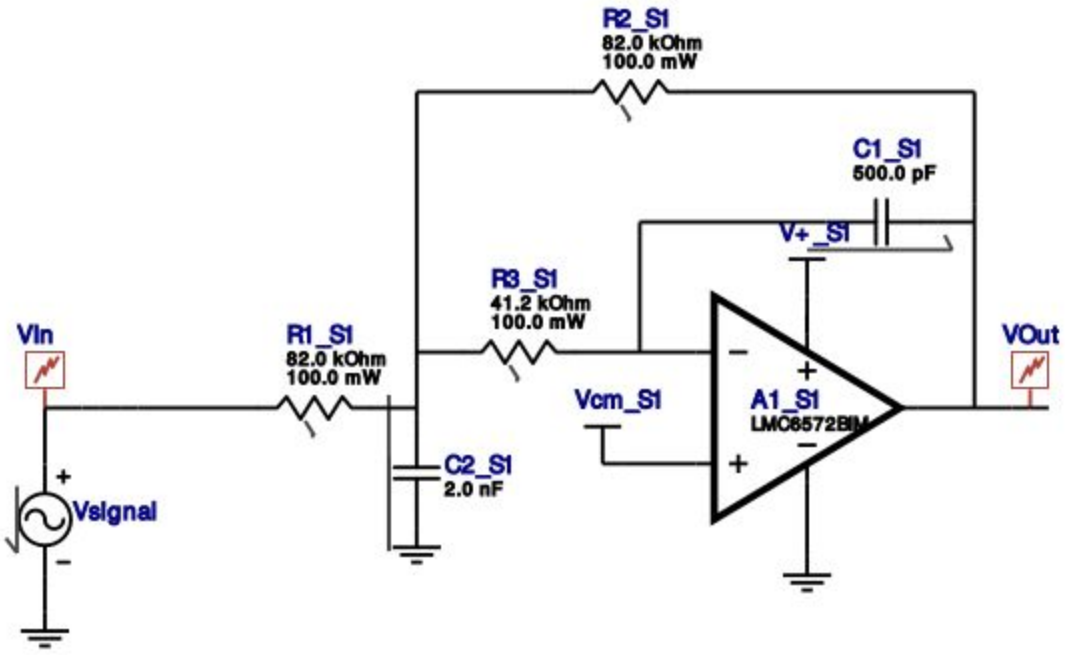
In this section, we will walk through our schematic design. Feel free to follow along here or using the image above. Please note that there are some errors with our current design that we will indicate. We will also offer steps that we took or could be taken to fix them. We will first walk through our analog design, and then move onto our digital design.

Our first consideration was how we would power our board, and how we would take in our input signal. We knew that the load cell with which we were interfacing would need to connect to our board at 4 points: two connections for power (5V and ground), and two to transfer the differential

input signal. Thus, we decided we would use an array of four 0.1” header pins to provide the necessary connections for our load sensor. Since the load cell wire connection were small, we determined that we would also be able to use the header pins designated for powering the load cell to be the power inputs for the entire board. Using simple header pins was ok, since our board should never draw much current. In future iterations, we may consider finding smaller connectors for the load cell and separate connections for power, however, size was not a significant concern to us in this project.

Next, we needed to condition our signal, first by converting it to single-ended and applying gain. To do so, we passed it through the LT1789 instrumentation amplifier, and, as explained in the Part Selection section above, set our gain to 33 with resistor R1. We did so since our signal ranged from 0-100mV, and the range on our analog to digital converter spanned approximately 3.3V. Therefore, we would be operating at full scale range, and thus get the most resolution out of our ADC. We wanted our scale to be able to sense “negative” force as well, since we know that, during impacts, the oscillations may cause the load cell to overshoot when decompressing, causing the negative input voltage to exceed the positive input voltage. Thus, we decided to offset our instrumentation amplifier by 0.5V in order to allow sufficient space for us to catch these negative components of the signal. We realize that our signal may not need this much room, but for an initial iteration it did not hurt to give it extra space. On the digital side, we would simply need to take into account that a voltage of 0.5V corresponded to 0 lbs of force.

For the anti-aliasing filter, our goal was to pass all signals at 1 kHz and attenuate signals at 50 kHz by a factor of at least a hundred (down >60 dB). We chose to use a Butterworth filter to achieve these constraints, and used TI’s online tool to design a single-stage butterworth filter that attenuated signals only .003 dB at 1 kHz and 73 dB at 50 kHz. TI’s tool let us play around with different designs before settling on the specific design, topology and component values (see Part Selection). The design we finally settled on is the following (as given by TI’s filter design tool).



The output of the anti-aliasing filter feeds into the positive input of the LT1864 analog to digital converter. Since our input signal was single-ended, we simply grounded the inverting input of the ADC. As mentioned above, we wanted the full scale range of our ADC to span 3.3V, since that is the range that our microcontroller can handle. Also, considering the presence of an offset on our signal, which allowed us to read “negative” force readings, we needed to add an equivalent offset to the supply of the ADC in order to avoid clipping within the expected signal range. Thus we used a resistive divider to create a voltage reference approximately equal to 3.8V.

We placed the level shifter in the middle of the connections between the ADC and microcontroller. We used pins PA08, PA09, and PA10 on the µC for serial clock (SCK), serial data out (SDO), and conversion enable (CONV), respectively. These pins were connected to the +5V side of the level shifter, and the corresponding partner on the +3.3V side of the level shifter was connected to the ADC. PA11 was chosen to sense signals from the output of LM311, but since the comparator’s HIGH output signal will be +5V, we also need to feed this signal through the level shifter. We chose PA02, an analog pin, to utilize the DAC in the SAMD21 and set the threshold value for the comparator.

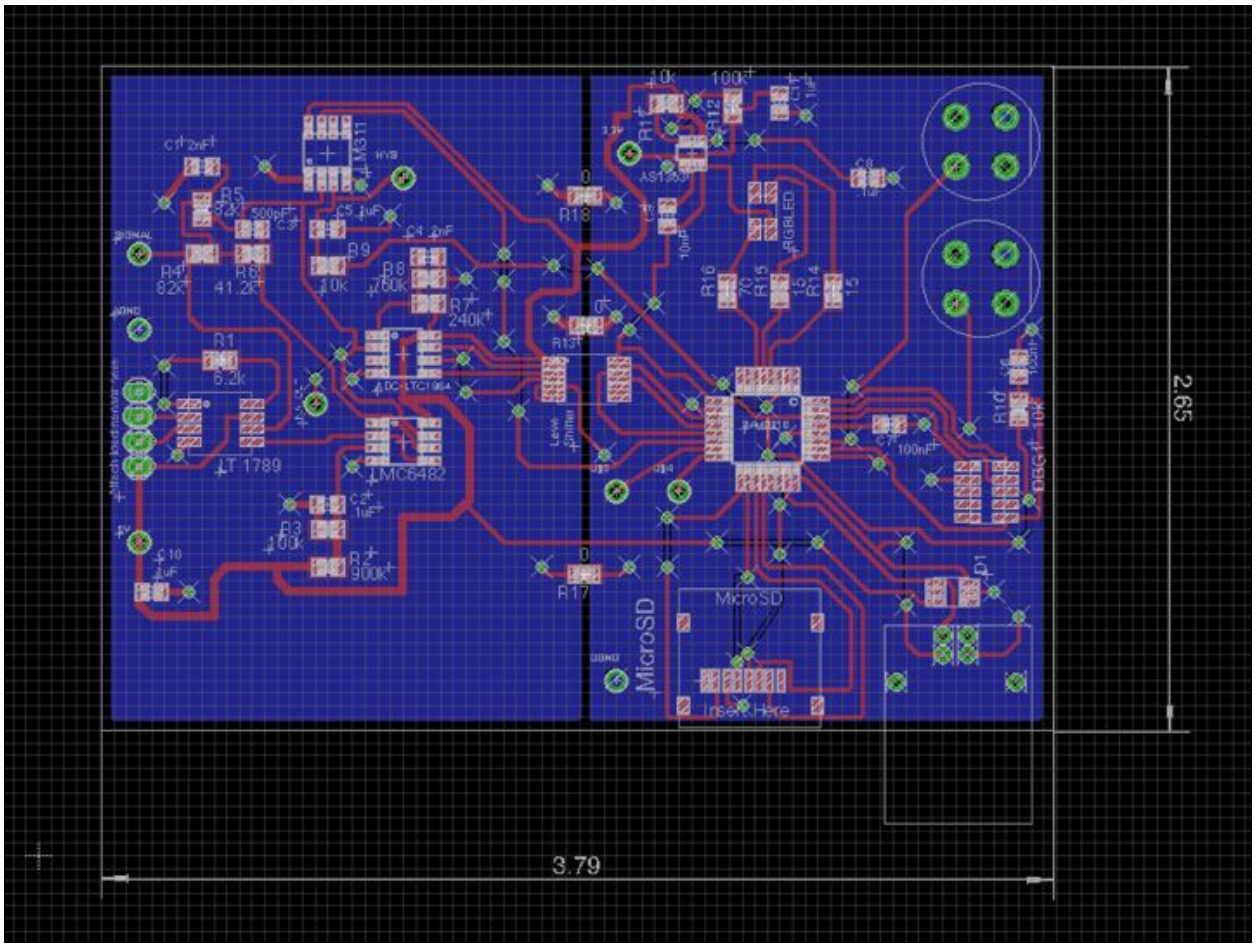
Having isolated one microcontroller pins that will be communicating with the analog side of the board, we used two sides of the microcontroller exclusively for digital purposes. We designated pins PA16-19 for connections to the micro-SD connector; PA24 and PA25 for communication with the USB; PA25 and PA27 for pushbuttons setups utilizing the internal pullups; PA26 for the reset button layout; and PA31 and PA32 to the debugger connectors for uploading code and

debugging. Near the threshold setpoint pin, we designated PA02-04 for each color of the RGB LED. Also necessary on the digital side was a diode array to prevent any overvoltage damage to computers attached to the board (We followed Avi's lead from the design for project 1 on this one).

We made sure to include test points at key places in our design: analog ground, digital ground, 5V, 3.3V, output of the instrumentation amplifier, output of the filter, and at two of the microcontroller digital pins for debugging purposes.

2.5 Layout

An image of our PCB layout, designed using the Eagle CAD tool is provided below.



Considerable care was taken in separating the analog components from digital components during the process of designing the layout of our board. As is clear from the image above, the analog (left) and digital (right) ground planes – highlighted in blue – are, for the most part, separated, but have 3 points of connection. This is to keep digital transients from inducing

current paths near the analog circuitry, which could introduce noise into our signal prior to its conversion to digital. We include points of contact between the two grounds to avoid the formation of unequal grounds and resulting ground loops. Since we had no size constraints for our board, we did not need to compact our design in this first iteration. We thus ran into few issues fitting all of the components onto the board and effectively routing the traces.

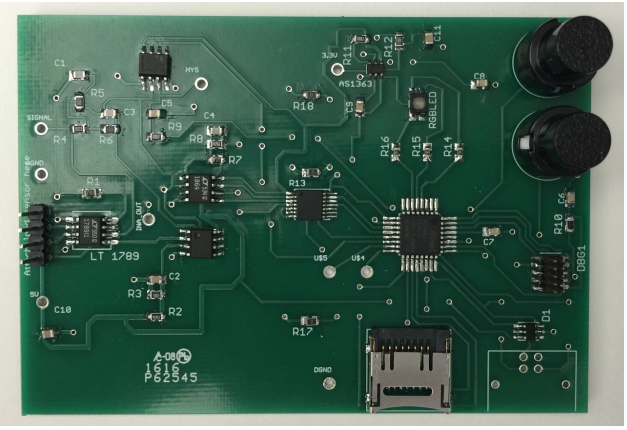
3. Operating Procedure

Although we were unable to make our board fully functional, we have outlined an operation procedure that would be functional had we been given more time.

- Calibration routine
    - Start by zeroing scale with “zero” pushbutton
    - Place known weight on scale, then press “calibrate” button
- Status RGB LED
    - Green: ready
    - Blue: event detected… sampling
    - Red: writing event data to SD card… DO NOT REMOVE card

4. Testing and Debugging

We began testing our circuit by making sure that the SAM-ICE debugger and the SAMD21E microcontroller could effectively communicate with one another. In between this project and the previous one, we corrected the pin connections on the debug header, so that this time around we ran into no issues programming our MCU, and soon after soldering the last component onto our board we were able to turn on and blink the RGB LED on our board.



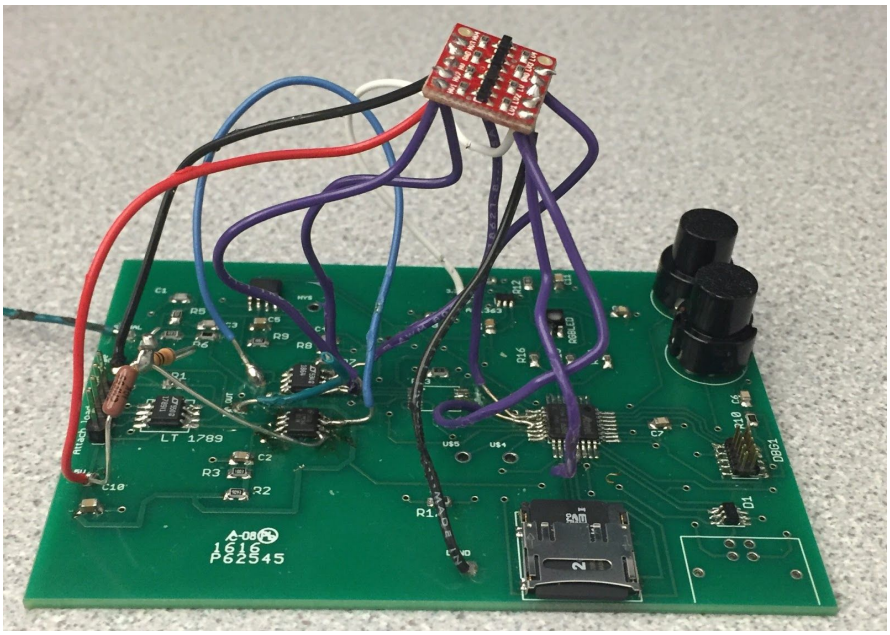
However, we found our first bug soon after. Since our microcontroller runs off of 3.3V, while our external ADC runs off of 5V, we needed to include a level shifter in our design, so that the two components could effectively communicate with one another. Unfortunately, despite spending a

significant amount of time searching for an appropriate bidirectional level shifter, we ended up ordering and assembling a unidirectional one. Therefore, we were unable to receive data from the ADC. Fortuitously, Professor Abrams had recently ordered a few bidirectional, 4 channel level shifters from Sparkfun, and he allowed us to use one.

After soldering wires between the microcontroller, ADC and the level shifter, we could effectively communicate with the ADC. However, first we needed to write the code for the job. Please see the firmware section for a description of the coding process.

We also noticed that our anti-aliasing filter was not working as expected. At first, we decided it would be better to put off dealing with the antialiasing filter, and attempt to write code that could interface with the ADC. Fortunately we had two well-placed test points that allowed us to easily bypass the filter. Once we got the ADC working properly, however, we needed to debug our filter.

So what was wrong with it? The filter seemed to be cutting off all values greater than the 0.5V offset voltage that we supplied to the non-inverting terminal. Well…of course it was! Our design was single supply, so if the inverting terminal voltage exceeded the non-inverting terminal voltage, the op-amp output would obviously rail. We were able to speak to Professor Abrams about the not so surprising phenomenon we were witnessing, and he confirmed that that was indeed the problem. So the error was in our design. In order to fix it, we needed to cut the trace that provided the 0.5V offset to the non-inverting terminals, and pull from a voltage divider between 5V and ground, that could provide 2V to the positive terminal, so that our signal would be guaranteed to fit within the full scale range of the ADC, and experience no clipping at the output.



After some debugging and rewiring, our board ended up looking something like this.

5. Firmware

Learning how to work with microcontrollers in an embedded system was perhaps one of the greatest challenges that we faced this semester. If the goal of the class was to frustrate us, we were thoroughly frustrated when it came to microcontroller programming. However, with some work, and with the help of Atmel's tool *Atmel Start*, we were able to write the code necessary to interface with our board's external ADC. We achieved a sampling frequency of approximately 2.94 kHz which considering the limitations of the M0-microcontroller is respectable.

In order to establish a fixed sampling frequency, we needed to provide a periodic clock signal to the external ADC's SCK and CONV pins. A timing diagram for the operating sequence of the LT1864 ADC, taken from the part's datasheet, is provided below.

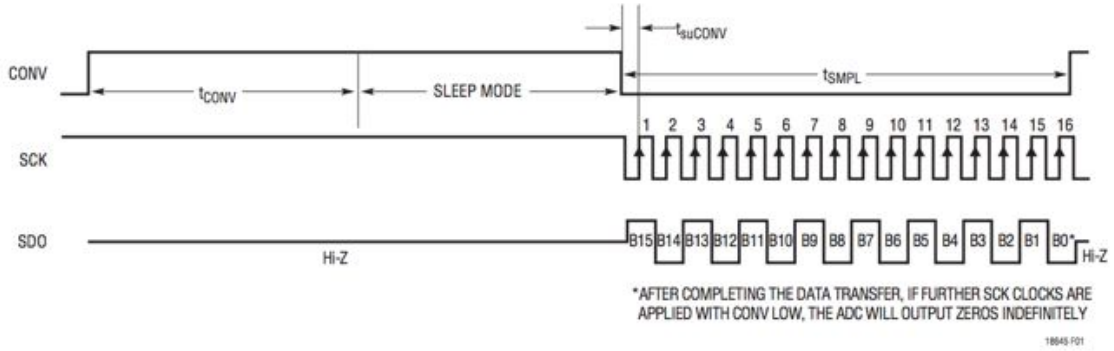


Figure 1. LTC1864 Operating Sequence

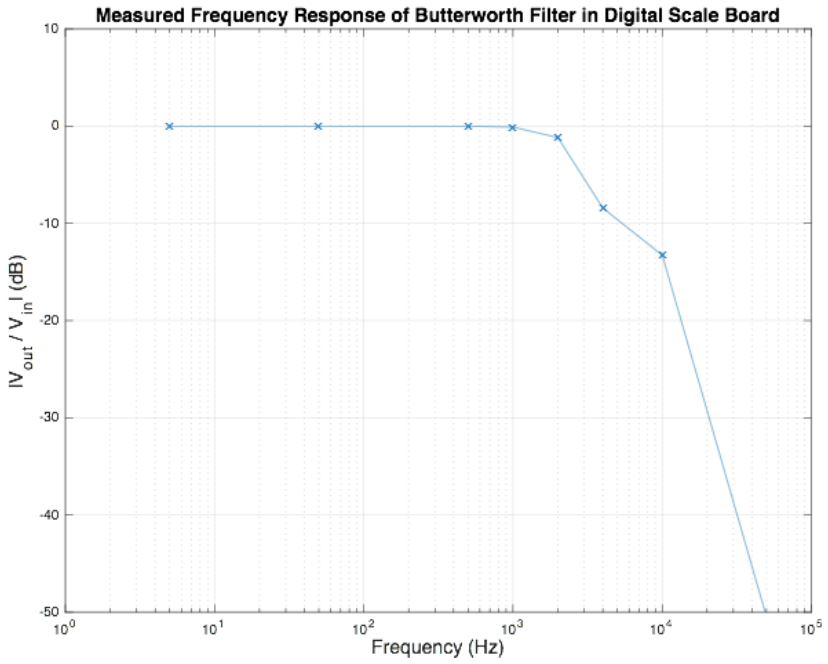
In order to establish a periodic signal, we utilized the timer peripheral of the SAMD21E microcontroller. We established 2 "timer tasks," which are called when a specified internal timer counter in the microcontroller counts up to some specified value. We made it so that the first task would be called upon the first count of the timer, and the second upon the second count. Within the first task, we simply pull CONV from low to high, in order to begin the conversion process. Then, one timer count later – allowing enough time for the ADC to finish its conversion process – the microcontroller enters the next timer task, which pulls CONV low, then reads data from the ADC by driving SCK and reading SDO upon each SCK rising edge. We store the data in an array of sixteen 1s and 0s – corresponding respectively to high or low SDO values –, and then call the function we named, "array_to_16bit()," to convert the data array to a 16 bit integer. This value is proportional to the voltage level read by the ADC. Given that it is a 16-bit integer, and that our ADC has an upper voltage level of 3.8V, the voltage level that the value returned by

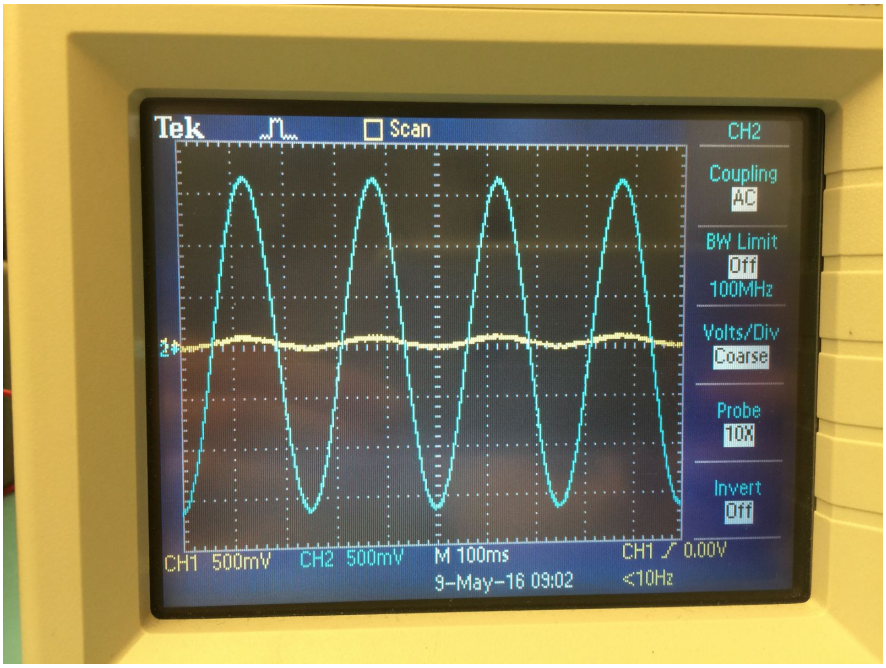
the ADC refers to is approximately equal to the ratio of the returned value to 2^{16} multiplied by 3.8V.

We sample for 1 second, storing each conversion value into a data buffer of an appropriate length. As of now, we have not yet configured our microcontroller to write to and read from an SD card. However, that would be the next goal for our project.

6. Results

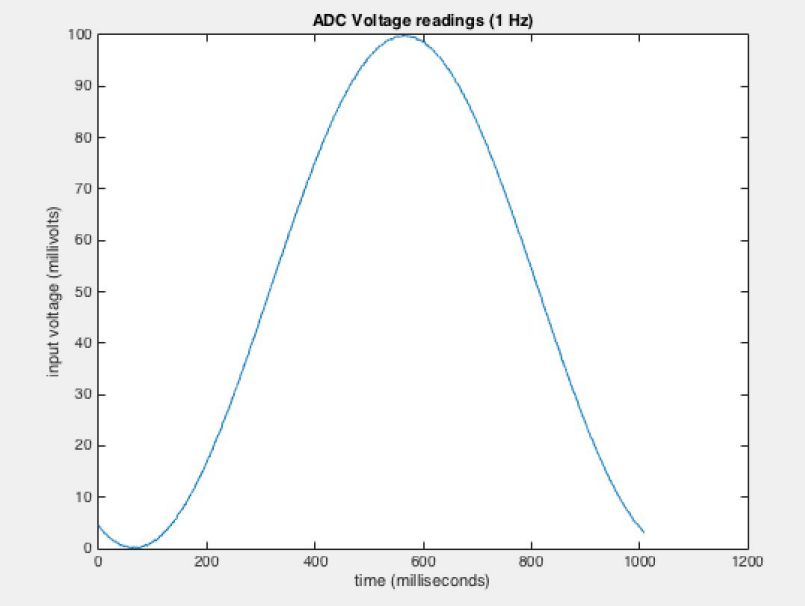
After we got the filter properly working, we tested its frequency response for a range of frequencies to verify it would attenuate signals outside of our sampling range. We wanted to simulate the signal from the strain gauge on the lab bench in a predictable way. Using the signal generator, we placed in a 100mV peak to peak sine wave (with a center offset of +2.55V) into the positive input of the instrumentation amplifier and a +2.50V DC signal into the negative input. Given the gain of the instrument amplifier, the output of the instrumentation amplifier was a ~3.3V peak-to-peak sinusoid. To get the frequency response of the filter, we scoped this 3.3V sinusoid as well as the output of the filter, and graphed the gain V_{out}/V_{in} for a range of frequencies. We found that the filter passed signals up to 1kHz within 1%, and then the gain quickly dropped off as frequency increased beyond 1kHz. While getting a fair amount of attenuation already at frequencies close to 1kHz was not our initial design, it was somewhat of a happy accident, since we found that once we started to program the SAMD21, it was difficult to sample much higher than 2.94 kHz. However, at 2.94kHz we did not end up having sufficient attenuation to eliminate the possibility of noise affecting our measurements completely. Also worth noting is that the instrumentation amplifier stopped providing the designed gain (G = 33) for signals above 2 kHz.

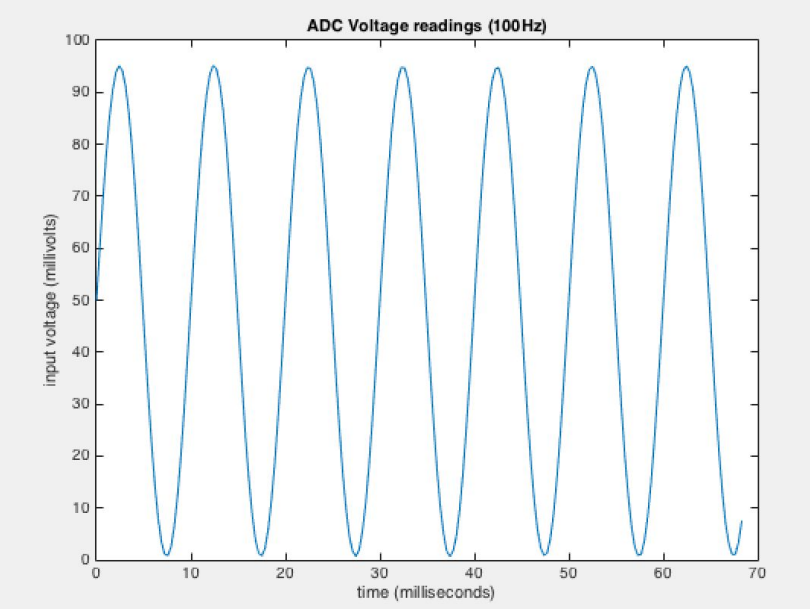


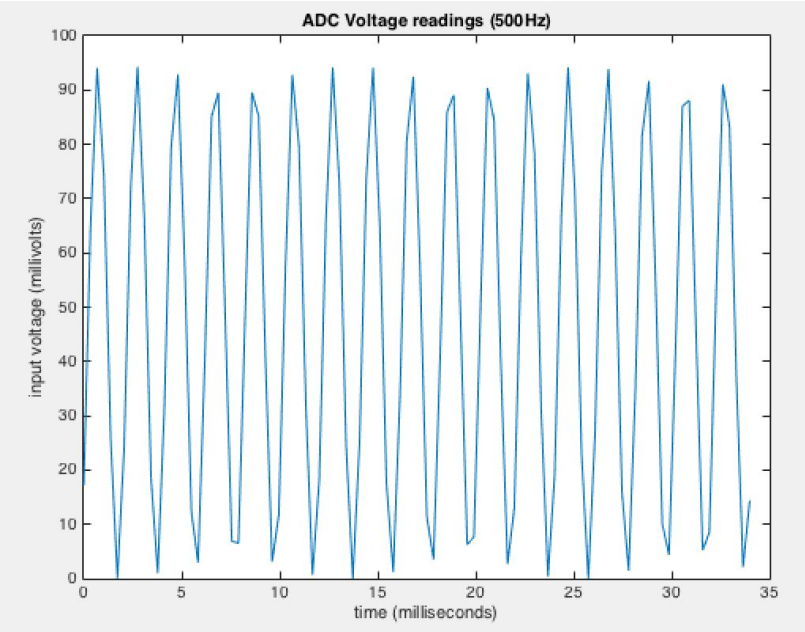


Having verified the filter would pass signals within the frequency range of interest, we needed to verify the performance of the ADC. To do this, we ran it for 1s and, using the J-Link debugger, were able to capture the values output by the ADC when a 100mV pk-pk sine-wave signal was passed into the input for 3 different frequencies: 1Hz, 100Hz, and 500Hz. We plotted the results we achieved for each frequency, and show them below. Please note that the x and y axes have been scaled so that the y-axis is of units volts, and the x-axis is of units time. Originally, the

y-axis was a 16 bit integer, ranging from 0 to (2^16 - 1), corresponding to the return value of the 16-bit analog to digital converter. We almost used the full scale range in our design: a 0V input signal corresponded to 4110 and a 100mV signal corresponded to 61,456 at the output of our ADC. Therefore, we lost about 4000 bits on either end.







As is clear from the graphs, we achieve strong conversion results for frequencies as high as 100 Hz. There seems to be little distortion in the signal at and below these frequencies, results which agree with the fact that we are sampling at approximately 2.9kHz. As you increase the frequency of the signal, however, it clearly begins to distort. This is expected, since our sampling rate is finite, and since we are not significantly oversampling. It is important to note, however, that our result still agrees with Nyquist, since the output signal still demonstrates the same 500 Hz frequency as the input signal. Yet our results indicate that if we want precision at high frequencies we would need to oversample, to avoid signal distortion.

We measured the ADC output value read by the SAMD21 for a range of signal values.

Voltage	Value measured by the uC from the ADC
2.49	65535
2.51	65535
2.53	56063
2.55	44479
2.57	32895
2.59	21423
2.61	13277

2.63	12543

7 Future Steps

There are a number of improvements to the design we would make in the second iteration of this project. First is a second set of wiring inputs for +5V power and ground, which would simplify connecting both the strain gauge and the power supply.
In RevB, we will design for lower gain on the instrumentation amplifier to make the single supply filter easier to design; given that we have a 16 bit resolution on ADC a slightly smaller input range on the ADC is acceptable. Also relevant to the single-supply filter, we will add a separate voltage divider to set the pseudo-ground of the anti-aliasing filter. We will also order the right bi-directional level shifter part. We will add more test points, since we discovered that during debugging one can never have too many test points. And finally, given more time to read up on Atmel documentation and to tinker in the lab, we would learn the programming necessary to make the board fully operational. The main steps necessary to utilize fully the microcontroller will involve: determining how to interface with SD card; developing code necessary to make the event detect comparator operational; and developing finite state machine for the calibration routine. Having verified the digital side works, we will test the board using the actual sensor. Although our setup mimicked the sensor output it remains important to verify this.

8. Appendixes

Datasheets:
LTC1798 Instrumentation Amplifier
LTC1864 16 bit ADC
LMC6482 Op Amp
LM311 Comparator
AMS1363 3.3V LDO
SparkFun Bi-Directional Logic Level Converter
SAM-D21 Documentation

9. References

1. Texas Instruments WEBENCH® filter designed. Retrieved from http://www.ti.com/lsds/ti/analog/webench/webench-filters.page
2. Final Schematic

10 Bill of Materials

Part Name	Qty	Value/ Precision	DigiKey ID	Manufacturer ID	Price @ Qty	Description	Eagle?
SAMD21E Atmel uC	1	n/a	ATSAMD21E18A-AUTCT-ND	ATSAMD21E18A-AUT	5.4	32kB SRAM… should be enough	Y
ADC	1	16-bit	LTC1864CS8#PBF-ND	LTC1864CS8#PBF	12.44	16-bit, SAR, analog to digital converter Max sampling frequency of 250ksps	Y
SD card	1	256Mb	1582-1117-ND	AP-MSD256ISI-1T	9.62	256Mb SD card	n/a
SD card connector	1	n/a	101-00303-68-1-ND	101-00303-68	1.26	SD card connector	Y
RGB LED	1	n/a	CLVBA-FKA-CAEDH8BBB7A363CT-ND	CLVBA-FKA-CAEDH8BBB7A363	0.55	LED to use for UI	Y
RED LED	1	n/a	160-1447-1-ND	LTST-C191KRKT	0.28	RED LED for power indication	N
Pushbutton	2	n/a	EG4791-ND	KS-01Q-01	0.53	Pushbuttons to interface with the Microcontroller	Y
Op-amp	2	n/a	LMC6482IMX/NOPBCT-ND	LMC6482IMX/NOPB	1.55	Precision Op-amp (LMC6482)	Y

.1” pin header						I believe that we have s few of these in lab… Needed for strain gauge-	Y
Instrumentation Amplifier	1	n/a	LT1789IS8-1#PBF-ND	LT1789IS8-1#PBF	7.41	Precision instrumentation amplifier (LT1789)	Y
Comparator	1	n/a	296-1388-1-ND	LM311DR	0.69	Comparator for Event Detection	Y
2X5 Header Pins	1	n/a	609-3695-1-ND	20021121-00010C4LF	0.80	Surface mount header pin strip for debugger (0.05” separation)	Y
Ferrite Bead						Still in the process of choosing value for this component	Y
USB micro connector	1	n/a	WM17144CT-ND	0475900001	0.81	USB micro connector (surface mount) We may want to check into this a bit more	N
Level Shifter	1	n/a	MAX3391EEUD+-ND	MAX3391EEUD+	2.44	Level shifter to shift 5V logic level from ADC serial pins to 3.3V logic level for uC.	Y
Capacitors	1	2nF	490-1627-1-ND	GRM2165C1H202JA01D	.20	Could only find +/- 5%... might need to look for better? This is for antialiasing filter	Y

Capacitor	4	510pF	490-11550-1-ND	GRM2165C1H511JA01D	.20	Capacitor for antialiasing filter… could not find 500pF, but figure that this may be good enough. We may want to test the capacitance of each individual to see which gives us a value closes to 500pF	
Diode Array	1	n/a	D1213A-02SO-7DICT-ND	D1213A-02SO-7	.41	Diode array for usb connector to protect computer	Y
3.3V Linear Regulator	1	n/a	AS1363-BSTT-33CT-ND	AS1363-BSTT-33	1.62	LDO Linear Voltage Regulator for MCU (+3.3V supply)	Y
5.36k resistor	1	5.36k	P5.36KDACT-ND	ERA-6AEB5361V	.63	.1%	
2.37k resistor	2	2.37k	311-2.37KCRCT-ND	RC0805FR-072K37L	.1	1%	
1.47k resistor	1	1.47k	311-1.47KCRCT-ND	RC0805FR-071K47L	.1	1%	
82k resistor	2	82k	P82KDACT-ND	ERA-6AEB823V	.63	.1%	
41.2k	1	41.2k	P41.2KDACT-ND	ERA-6AEB4122V	.63	.1%	

Ferrite bead	2	220ohm @ 100MHz	311-1.47K CRCT-ND	490-1054-2-ND	.1	Ferrite Bead for Gnd plane connection	