System Integration of BioHaptix System for Tissue Impedance Measurement

A Design Project Report

Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering, Electrical and Computer Engineering

Submitted By:

Shiva Vignesh Rajagopal

MEng Field Advisor: Bruce Land

MEng Outside Advisor: Jonathan Butcher

Degree Date: May 2016

Table of Contents

Abstract

Master of Engineering Program

School of Electrical and Computer Engineering

Cornell University

Design Project Report

Project Title: System Integration of BioHaptix System for Tissue Impedance Measurement

Author: Shiva Rajagopal

Abstract: Soft tissues are complex materials whose mechanical stiffness and electrical impedance characteristics are extremely useful in a variety of ways, some of which include measuring the likelihood of pre-term birth. Dr. Jonathan Butcher, along with previous students, has developed a device for mechanical and electrical impedance measurement, using a single device that can vary pressure while sending a variable-frequency electrical signal through the tissue. This device can be used for nondestructive, fast, quantitative measurement of soft tissue, which can be used for diagnostic purposes. This design project aims to integrate two different versions of this system, including fixing errors, making the systems more resilient, and centralizing information and documentation.

Executive Summary

Characterizing soft tissue in living beings is an absolute necessity for a variety of application, including health, disease, and healing. Soft tissue exhibits natural stiffness and impedance characteristics that represent its structural composition, cellularity, and other important qualities. As a result, the ability to characterize this tissue without excising it is critical, as excising the tissue disrupts these characteristics, is destructive to the host, and creates measurement error. Dr. Jonathan Butcher and many students have worked on a system that can accomplish real-time characterization of the mechanical and electrical properties of soft tissue.

The system works by two main physical systems controlled by a desktop computer and microcontrollers. A Windows application controls both the microcontroller-based function generator, which sends a specified frequency through to the probe. The Windows application also controls an Arduino, which controls the vacuum system and triggers the oscilloscope to read data returning from the probe. The data is acquired through a PicoScope 4226, which is a computer-connected oscilloscope, and then analyzed by a MATLAB program.

There are two versions of the system that have been developed over a number of years. The first is housed in a computer tower, and is meant primarily for development. However, there are many parts of the development system that are not soldered well, have loose connections, or are poorly integrated. Additionally, the documentation for this system was not complete. To alleviate this problem, Dr. Butcher's Lab acquired a newly integrated version of the product from a professional engineer, using the same components, but in a quieter, more tightly integrated package. However, due to the lack of proper documentation, the new version of the system was not functional, lacking some important parts and containing some incorrect or broken parts.

My original contribution to this project was intended to be creating a PIC32 microcontroller-based replacement for the expensive PicoScope and perhaps integrating it together with the PIC32-based function generator. [3]. However, due to the arrival of the broken second version and the need for both versions of the device to be more resilient and functional, the scope of the project changed to integrating both versions of the device on all levels, including hardware, firmware, and software. Additionally, throughout the year I developed a prototype of a PIC32-based data acquisition device based on an external, high-speed analog-to-digital converter. This prototype is not ready for integration with the system, but is a good step towards drastically cheapening the cost of the system and can be picked up by future students.

By the end of the year, I was able to improve the reliability of both systems and provide corrections and improvements to the documentation of the old forms of the system. The software became centralized between both systems and presents a much better interface to a user. The V2 system has been functionally verified and fixed from its original, incorrect form, although it needs to be verified against actual tissue and gel tests, since the probe is not reliable at this point. I was also able to assist the Biomedical Engineering team in creating a new edition of this product with much newer technology. Between the two old systems and the new one created by the BME team, the original vision of this project is closer than ever to becoming a real, marketable product.

1. Introduction

1.1. Soft Tissue Characterization

Characterizing soft tissue in living beings is an absolute necessity for a variety of applications, including health, disease, and healing [4]. Soft tissue exhibits natural stiffness and impedance characteristics that represent its structural composition, cellularity, and other important qualities. As a result, the ability to characterize this tissue without excising it is critical. Testing the tissue *in vitro*, or outside its biological context, can lead to a lot of experimental error, as described in [2]. To alleviate these issues, Dr. Jonathan Butcher and many students have worked on a system that can accomplish real-time characterization of the mechanical and electrical properties of soft tissue. Using a pipette-aspiration (PA) system, as described in [1], the system tests the soft tissue's mechanical properties *in situ*, or in its original place, in a minimally invasive fashion to preserve the tissue properties as much as possible. While the tissue is in a vacuum, it is easy to send an electrical signal through it to measure the electrical impedance as well. Tarsi et al. show in [5] that this is an accurate, high-speed method of testing tissue.

The device that has been developed by Dr. Butcher's lab and BioHaptix Inc. is an implementation of this design, which we can visualize with Fig. 1. The fixed vacuum will pull in the tissue to the extent that it can, and using the number of electrodes that it touches (visualized by the colored blocks on the sides), we can determine its mechanical characteristics. Additionally, while the tissue is inside the probe, we can run an electrical signal across it to get the electrical impedance.

1.2. BioHaptix Systems

There are two versions of the machine that are currently in use by Dr. Butcher's Lab. The first, known as V1, is a development unit that is housed in a computer tower. All of the system components are housed inside a computer tower, as shown in Fig. 2, but all connections are meant to be easily swappable and accessible. The system itself, unfortunately, was extremely fragile and not well-engineered, so functionality was almost non-existent. The second system has exactly the same hardware as the first but in a more integrated design, courtesy of Dave Adams, as shown in Figs. 3 and 4. The V2 system was intended to be a more reliable version of the V1 system, but due to incorrect information, V2 arrived missing some parts and containing some incorrect components. The primary focus of this project was fixing these two systems and making them functional.

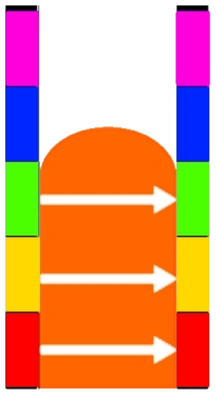


Figure 1: Probe Tip Model

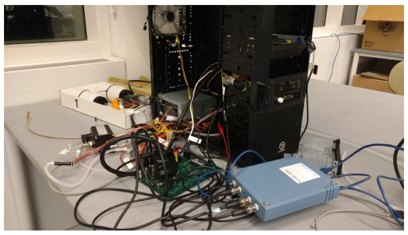


Figure 2: V1 System

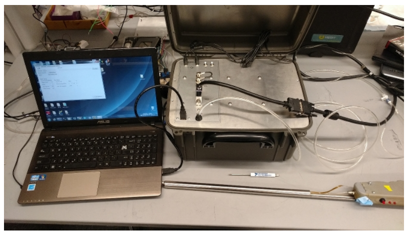


Figure 3: V2 System

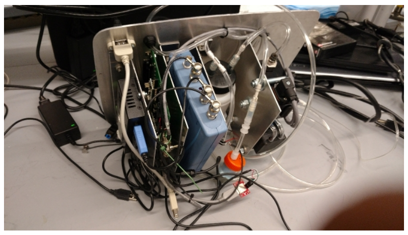


Figure 4: V2 System Inside

The secondary objective of this project was to replace the PicoScope component for data acquisition. The PicoScope 4226 which is currently in use can sample at 125 megasamples per second at a resolution of 12 bits, which is more than enough for this system, which only ever uses frequencies of maximum 100kHz. However, it also retails for $1,149. To help improve this, I created a prototype for a new system that uses a small, inexpensive PIC32 microcontroller and an inexpensive external analog-to-digital converter (ADC) to mirror the PicoScope's functionality. The sampling rate of the ADC is far lower, at 3 megasamples per second at 14-bit resolution, but this is still more than enough to perform the necessary calculations for impedance.

In this paper, I will first discuss the current system design in detail in Section 2, primarily as a reference for future users and developers of the V1 and V2 devices. Next, I will discuss my contributions to the various aspects of the project in Section 3. I will then discuss the results I was able to obtain in Section 4 and provide some concluding thoughts in Section 5. Some additional information and code is located in Sections 7 and 8.

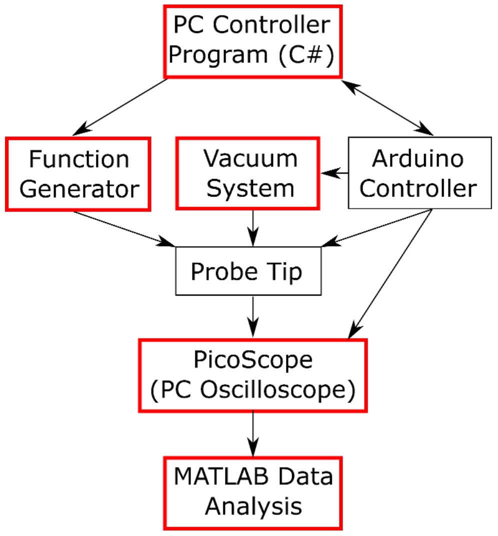


Figure 5: System Block Diagram

2. Current System Design

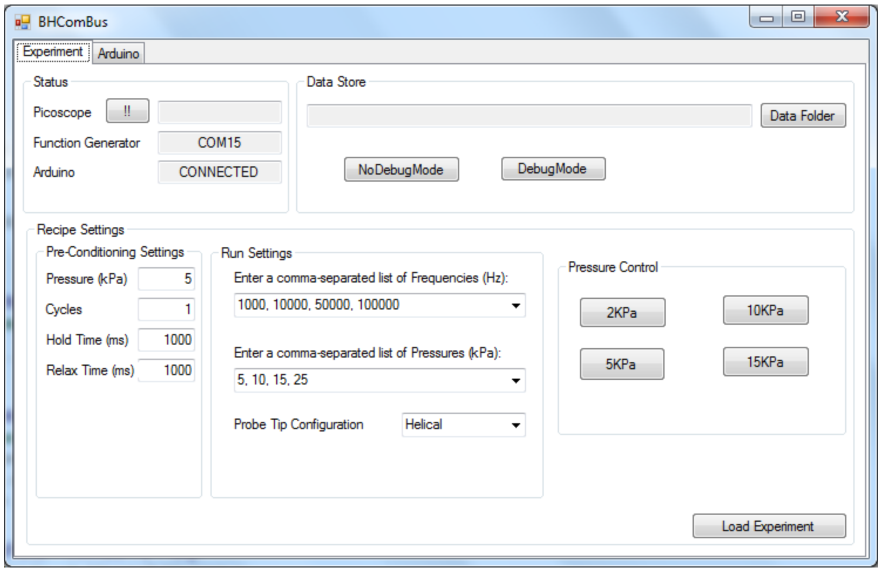
A block diagram of the entire system's dataflow can be seen in Fig. 5, with the parts that I affected in this project outlined in red. In this section, we will detail all parts of the system in enough detail that a newcomer to the system should be able to understand the underlying high-level communications and commands that this project involves.

2.1. PC Controller Program

The PC Controller program, known as BHComBus, is a Windows Forms application written in the C# language. The app's primary usage is setting up the experiment, and preparing all other parts of the system for the experiment as well. A screenshot of the system can be seen in Fig. 6.

The application provides a two-tabbed interface. The first tab is for setting up the experiment, including the preconditioning settings. The preconditioning settings are for exercising the tissue before the actual experiment begins, since the tissue will not be in its equilibrium state immediately. The user can specify the number of preconditioning cycles, as well as the pressure and cycle times for the preconditioning. After, the user can enter a list of pressures to experiment with, as well as the frequencies of the electrical signals that should be tested at each pressure. Once the user has selected these parameters, they can press the Load Experiment button and flip to the Arduino tab of the program. The Arduino tab allows the user to keep an eye on the experiment, as well as read messages from the Arduino controller, such as when the experiment is complete and it is safe to remove the tissue from the probe.

The app also has the ability to select the two serial port numbers for the function generator unit, as well as the Arduino unit. Both units should be plugged in before the program is started, but if the program cannot find either, it will prompt the user to do so. The program's code contains



Picoscope !!
Function Generator COM15
Arduino CONNECTED

Pressure (kPa) 5
Cycles 1
Hold Time (ms) 1000
Relax Time (ms) 1000

Enter a comma-separated list of Frequencies (Hz):
1000, 10000, 50000, 100000

Enter a comma-separated list of Pressures (kPa):
5, 10, 15, 25

Probe Tip Configuration Helical

Figure 6: BHComBus App

all the dependencies (including the dynamic link library and the drivers) necessary to control the PicoScope, but this functionality has not yet been implemented.

2.2. Function Generator

The function generator module is a simple PIC32 microcontroller with a resistor ladder that enables it to generate a variety of sine wave signals. The function generator was developed by Mingyuan Huang (M.Eng '15) and Bruce Land. The PIC32 is powered through the Microchip Microstick II board/programmer, which gets its power from a mini-USB plug connected to the system USB hub. The function generator is configurable at runtime through a UART channel at 57600 baud, which the user can independently test using PuTTY or another terminal emulator. The BHComBus program connects to the function generator upon starting the program, and can quickly switch the frequency at runtime.

2.3. Vacuum System

The vacuum system consists of a vacuum pump, two reservoirs for pressure, a pressure regulator, two solenoids, and a feedback transducer. The vacuum pump creates the pressure in the system, which is stored in one reservoir. The other reservoir serves as a muffler for the pump to keep the system as quiet as possible. During the experiment, the pressure regulator (ProportionAir), lets some of this reservoir pressure through to the probe. To control the flow from the pump to the reservoir, and from the regulator to the probe, the two solenoids can be controlled by the Arduino. The vacuum system has some noise cancellation hardware on it, such that the system can remain as quiet as possible. The feedback transducer feeds back an analog voltage representing the pressure that it senses back to the Arduino, which uses its onboard ADC to convert this to a pressure value. The probe also has a pressure transducer inside, which is identical to the transducer on the reservoir.

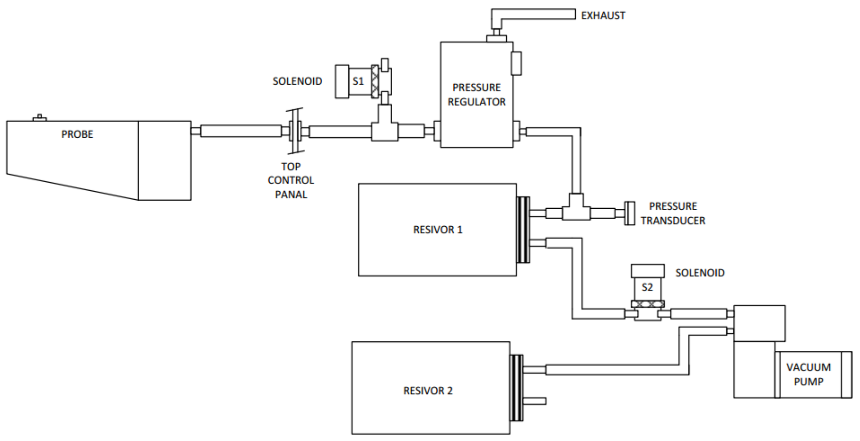


Figure 7: Vacuum System (from Dave Adams)

2.4. Arduino

The Arduino is the central controller of the experiment. The probe's components are primarily connected to the Arduino, including the vacuum control, vacuum feedback, mux control, PicoScope trigger, and interface with the BHComBus app. The Arduino communicates with the BHComBus app during experimental setup to receive the electrode pairs that it must test as well as how to precondition the tissue. The button on the probe commands the experiment to start, during which the Arduino adjusts the pressure of the system and commands BHComBus to change the frequency to the next setting in the experiment. Once this is complete, the Arduino triggers the external trigger on the PicoScope to acquire the data for the current electrode pair. The Arduino, like the function generator, is controlled by a UART connection at 57600 baud. There are different debug modes in the Arduino code, which are generally helpful for being able to see various data as the experiment continues, although different modes can be problematic. See section 3.2 for more about how this was used and some quirks while using it.

2.5. Probe

The probe is a 3D printed assembly holding a custom PCB, a pressure sensor, and all the necessary circuitry for interfacing with the rest of the system. The probe interfaces with the system through a DVI connection with many connections, including the function generator signal, power for the board, and the mux selection lines. The probe board does not have a proper connector on it, instead electing to connect all the wires with through-holes. There is a pressure sensor on the probe board itself too, which reports the pressure at the actual probe tip back to the Arduino for feedback purposes. When the experiment is in progress, the Arduino makes use of the two muxes on the probe board to scan through all electrode pairs. There are two 16-way muxes on the probe board, each connected to eight pins of the spring loaded connector. The input mux takes in the function generator signal and the Arduino selects which electrodes to send and receive the signal with by toggling the mux select lines. The flex circuit connects to the spring loaded connector when the probe door closes, and all electrodes are connected to their respective muxes. Since the Arduino knows which electrodes correspond to each other in the flex circuit, it can ensure that if there is tissue on the probe, the signal is fully passed through to the PicoScope. When the Arduino is toggling the mux, it also sends a trigger signal to the PicoScope to trigger synchronized data collection. Each mux only uses 8 of its 16 connections, which means the circuit could be simplified in the future. A schematic of the probe board can be found in Section 7.

2.6. PicoScope

The PicoScope 4226 is a computer-connected oscilloscope that can run at 125 megasamples per second with 12 bits of resolution. The PicoScope connects with the laptop using a USB connection, though it needs a good amount of power. Thus, if using the PicoScope with a USB hub, it should always be a powered hub, or a hub with USB 3.0 support. The PicoScope's channel A is connected to the voltage signal that is running in to the tissue and should be visible on the trace. Channel B is connected to the returning current signal from the tissue. These are all connected via BNC connectors from the main board, returning from the probe. As mentioned before, the Arduino connects to the external trigger port on the oscilloscope to synchronize the data capture with the electrodes. The data is visualized and exported using the PicoScope program in Windows, which enables the user to visualize the data and save it in a variety of formats. The PicoScope comes with a DLL (dynamic-link library) that enables it to be connected with an external program, such as BHComBus, but this has not yet been implemented. The simplest way to move forward with this system's software would be to set up the PicoScope to deliver its data into the BHComBus program directly, and run the impedance calculations using this data. Currently, however, the data is saved into text files and analyzed by MATLAB. There is one text file for each oscilloscope trigger, which can easily end up with a single test producing hundreds of text files depending on its complexity.

2.7. MATLAB Data Analysis

The MATLAB Data Analysis program was written during the 2014-2015 year by Alex Kim. The program reads in all of the text files generated by the PicoScope program and then analyzes the data, including stress and strain as well as impedance. Some experimental parameters must be input before the test can run. The process is fully automated, though not fully verified.

2.8. Infrastructure

These parts of the system warrant some explanation, just to know the system better and understand its underlying infrastructure, including dataflow and power.

Power

The mix of all the various components in this system creates some unique power requirements. The vacuum system creates a lot of electrical noise, and thus there is a need for it to have its own power supply. Thus, one power supply provides power to the vacuum system alone, while another supply controls the rest of the components, including the probe, the main board, and the USB hub. The V1 and V2 systems contain different ways of dividing this power, with V2's method being much cleaner and more reliable.

USB Hub

The USB hub is a 4-port USB 3.0 hub. The primary purpose of this is distributing the power to the smaller components and providing connectivity to the microcontrollers and the PicoScope. The PIC32 function generator requires two USB connectors: one is for powering the generator, while the other is for connecting through the UART connection. The Arduino is powered through a pin, and its UART connection runs through the hub. Finally, the PicoScope draws its power and data connection from the USB hub. There should be no need to pay attention to which USB port each component is plugged in to.

V2-Specific Motherboard

The V1 system uses different power supplies and power divison methods, as well as distributed circuit boards for the various subsystems. The V2 system, on the other hand, combines the power division, the function generator components, and the Arduino/base board components into a single PCB with all necessary connections. This makes everything far more reliable, as its connections are professionally engineered.

3. Contributions to the System

My contributions to the project were distributed among virtually all parts of the system, including debugging and development. As mentioned before, the block diagram in Fig. 5 contains all components that I affected in this project outlined in red. In this section, I will detail all of the contributions I made to the various parts of the system.

3.1. Version 1

The primary issue with the V1 system is its unreliable connections. The wires and connectors were soldered together poorly, and when connections seemed weak, it appears that previous students used ad hoc solutions like electrical tape and duct tape. During the 2014-2015 year, the M.Eng students were able to test the system on a horse, but the V1 system suffered damage from the transportation, leading to a virtually broken system. V1 was deemed to be a slightly less worthwhile effort than V2, since V2's connectors were far more reliable and the focus could be more on debugging the actual implementation errors with the system. Thus, my work on V1 was limited to a few sprints, each focusing on a different part of the system.

The first effort to fix V1 involved the main board connectors. The connectors to the main were soldered unreliably, and did not have heatshrink around the critical points in the system, leading to some wires being shorted together. One connector in particular was desoldered, but was also covered up by heatshrink, a layer of electrical tape, another layer of heatshrink, and a layer of duct tape. With the help of Jack Thompson, I was able to make some of these connectors more reliable, though some work still needs to be done to make them all trusted connections.

The second effort with the V1 system involved fixing the vacuum system. The ProportionAir pump was found to be a point of failure in the system, and when its connector was opened, all of the wires had come apart from their solder. After re-soldering this connector, the vacuum system was still not working due to the Arduino not being programmed to control the pump correctly. At this point, the vacuum system on V1 has not been fully verified, but the connections are far more reliable than they ever were.

The third effort with the V1 system was focused on the case and power connectors to all components. The main power supply runs all of its power division through a traditional PC power supply, but its connections to the base board were also coming desoldered, so these were reinforced. Additionally, the vacuum's power supply ran through a barrel connector in the back of the case, but its wires

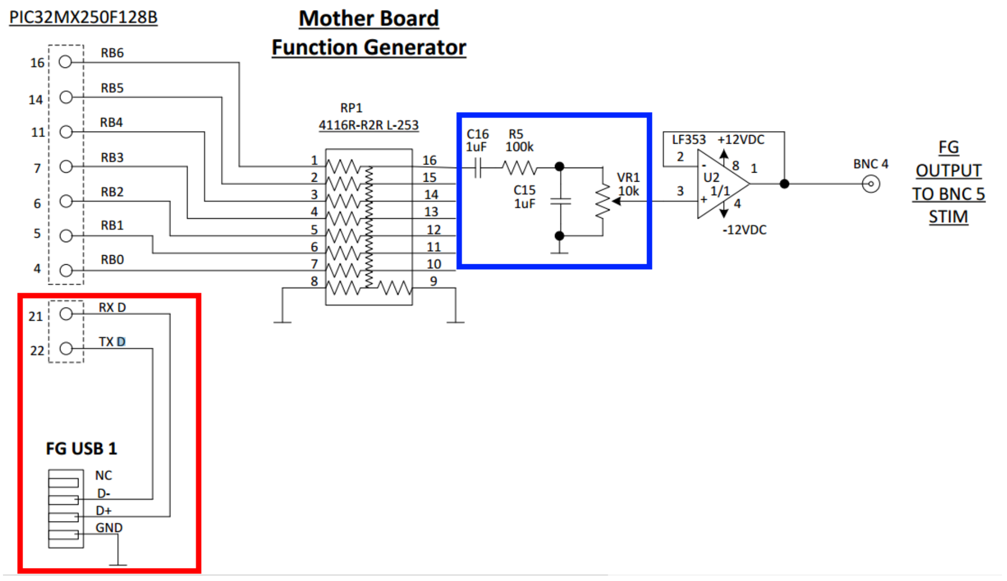


Figure 8: V2 Function Generator Schematic

were also coming disconnected from the barrel connector and the vacuum system, so these were reinforced as well.

3.2. Version 2

The V2 system is where most of my system integration has been centered. When I came on to the project, the new V2 device had just arrived, and I was asked to flash the Arduino and the PIC32 with their code to check if the system was functional. What followed, however, was a large, concerted effort to learn how the system worked, why certain parts of the system were not operating as expected, and what design errors led to the V2 device not functioning.

The first flaw I noticed almost immediately concerned the function generator on the V2 board. As seen in the red box in Fig. 8, the UART RX and TX pins from the microcontroller are connected directly to the two data pins on the USB, which are not compatible. The V1 system used a USB cable with a built-in FTDI USB-UART chip, which was documented in [3], but did not make it to the final V2 system schematic. Thus, the wires were directly connected. The USB connector is a USB type A connector, which connected directly to the USB hub. To alleviate the issue, I used a SparkFun BOB-12700 USB Type A Female Breakout to gain access to the pins, which I then connected to a USB-UART cable and plugged into the hub. Using PuTTY, I was able to verify that this workaround was successful. I also updated the documentation for the motherboard to include this in the future.

The next step in debugging the V2 system was also regarding the function generator. When attempting to unit test the function generator, I discovered that the signal was extremely low voltage, far too low for a good signal-to-noise ratio. Comparing the documentation for the original system, created by Mingyuan Huang and Bruce Land, with the V2 schematic [3], I noticed that one capacitor was incorrect, resulting in the system attenuating some of the frequencies we intended to use. This is C15 in the blue box shown in Fig. 8. This capacitor was initially 100pF, but unfortunately a 1µF capacitor was used instead. To alleviate this issue, rather than wait to order many 100pF capacitors, I decided to use two 22pF capacitors that were on hand, and solder them to the board in parallel (in a stacked formation). Once this was done, the function generator worked as planned.

The next task on the V2 system was focused on the vacuum system. The pressure created by the vacuum pump seemed to be never reaching the probe, and although pressure was building up in the reservoir, the pump was continuing to create more vacuum. The debugging for this process proceeded by individually testing the solenoids and the pressure at each point on either side of them. It was eventually uncovered that the pressure transducer to measure the reservoir pressure was faulty, and was constantly reporting low pressure. The system would not let pressure through to the probe until it achieved its set point in the reservoir, and thus the pressure could never be felt at the probe tip. Attempts to debug this sometimes involved the various debug modes already on the Arduino, although during experimentation, the "`DEBUG_ALL`" mode would not run the experiment, possibly due to a saturated UART channel.

There was a brief issue with the noise level of the system during the later part of the year. Turning on the vacuum system was causing a lot of vibration from the vacuum pump, making the entire system unfit for testing on animals, as this could startle the animal. With the help of Jack Thompson, this issue was alleviated by inspecting the vacuum pump and giving the top level of the vacuum a clean, we were able to drastically decrease the noise level in the system and once again make it suitable for use.

The rest of the integration effort on the V2 device was primarily focused on the probe. Dr. Newton de Faria and Nariman Ziaee attempted earlier in the semester to figure out whether the probe was operating as expected, but unfortunately their conclusion was to abandon the probe. There is only one functional probe and it is still unknown whether it actually functions correctly. It has not yet been tested whether all of the circuitry is actually functional, although the probe's pressure has been found to be working correctly. There is a push connector connecting the tubing from the probe's box to the external system which was found to be somewhat weak, but it is workable for now. The AC signal has been traced all the way to muxes using an oscilloscope, but it has not yet been confirmed whether the muxes are working correctly, nor whether the signal returning from the system is what we expect. However, it is absolutely necessary to redesign the probe. The current set up connects the 2x8 flex circuit to a spring connector of the same dimensions, but the actual connection of these two is terribly inconsistent. Once the door closes, it is very easy for the connector to slip, thereby ruining the entire circuit. Additionally, just the process of opening the door leads to the flex circuit being ejected. A more reliable method of connecting the flex circuit needs to be developed for this to be a reliable product. One possibility is a modifying the flex circuit to use a flat flex connector that would both remedy the complication of having a 2x8 connector, and provide a more secure connection. An example can be found at here. Additionally, there is virtually no reason why all of the wires from the main board need to be soldered with through-holes into the probe board, and this could easily be replaced by a conventional DVI connector, making the probe board itself more modular and much easier to create. From a usability standpoint, V2 should be the only system that is used to test on tissue and animals, as it is the only system whose hardware has been verified. The probe has not been fully verified at the time of writing.

3.3. Software

The software was one of the most difficult parts of this project to remedy, with three different versions of the software and almost no indication of the differences between each one. Additionally, all versions of the software contained a significant amount of obsolete code for the old function generator and for old features of the system. My goal with the software was to centralize the code among all versions, remove the obsolete code, and add some features to make the software more resilient.

The first step was scanning through the code and removing the code for the old function generator. The old function generator was a Tektronix function generator that could be controlled by a computer, but took an excessively long time to switch frequencies, and was very large. The PIC32 simplified both the circuitry and the interface, and sped the entire system up by a large factor [3]. I removed all obsolete code and interface objects from the old function generator, decreasing the code size and overall size of the project. This improves usability and reduces confusion for new users and developers of the system.

The next main step of the code was creating a more intelligent method of finding the serial ports for the Arduino and function generator. In its original state, the program had the ability to choose the Arduino COM port at runtime, but the function generator was hard-coded in the code at compile-time, which would not be acceptable. Additionally, the program was reading the list of possible serial ports from a Windows registry location that contained every COM port name the system had ever used, rather than only the names that were currently in use. Based on another location in the registry, we could see the list of active COM ports, and based on knowledge of the hardware and the manufacturers of the serial chip, I integrated a better heuristic for finding the COM ports for both. This was not a full solution, however, as the specific serial converter used for either controller might change, and the heuristics may not always be valid. The runtime selection of the Arduino COM port is optimal for this case, but is very complicated to duplicate for the function generator. However, I realized I could take advantage of the fact that the COM port name should not change after starting the program. For this reason, I created a method of selecting the function generator COM port upon startup of the program in case the heuristic fails.

One of the most important features of the system is the ability to cycle through multiple frequencies at multiple pressures. However, the software had no method of inputting multiple frequencies or specifying the exact pressures to test at. To alleviate this issue, I implemented a modified user interface that includes the ability to input a comma-separated list for both pressures and frequencies. Additionally, I added the ability to have some preset lists for common experiments. This makes setting up the experiment drastically easier, and is finally no longer dependent on hard-coded values and code.

My final contribution on the software side was fixing the MATLAB code. To analyze the data coming from the PicoScope, the MATLAB code had to take in all of the data and calculate the impedance of all data with knowledge of the underlying frequencies and pressures used. However, after combing through the code, I found that it was not compensating for the known resistance introduced by the system itself, as well as containing an incorrect equation that was making the results incorrect by a power term. After modifying the script and recompiling it in MATLAB, I was able to verify that the script was outputting reasonable results based on the original data and experimental parameters that I found on the laptop.

3.4. Data Acquisition Prototype

A significant amount of time was spent this year on attempting to create a system that could replace the $1,149 PicoScope. The maximum frequency we are using with this system is 100kHz, so a sampling rate of 125 megasamples per second is far more than what we will ever need. This was the main project that I expected to complete this year, although this quickly turned towards

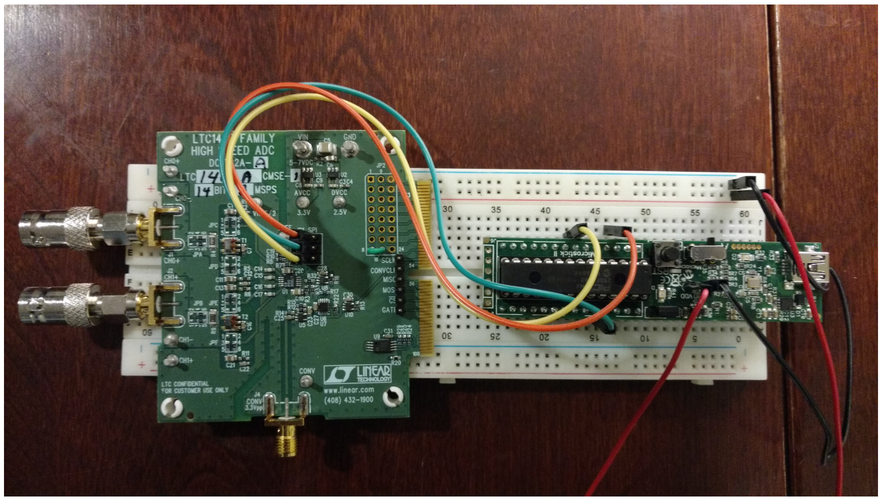


Figure 9: Data Acquisition Prototype

debugging and system integration of the two systems. I came in with the expectation that I would use a PIC32 microcontroller as the main computing power for the system, as the function generator was already based on a PIC32, and if possible, I could integrate the function generator and the data acquisition together into a single microcontroller.

The next step was determining which analog-to-digital converter I would use for the data acquisition. The PIC32MX250F128B I planned to use has an onboard ADC that can acquire 10-bit data at 1MHz using direct memory access (DMA). While this sample rate would be sufficient, this application requires two channels to be acquired simultaneously, and the latency in switching ADC channels would make the sample rate insufficient for this system. Thus, I determined the best solution would be an external ADC. After some research, I came upon the Linear Technology LTC1407 series of ADCs. These ADCs could sample with a 2.5V range, contained common mode rejection, and had extremely high accuracy at 14 bits. The sample rate was 1.5 megasamples per second for each channel, for a total of 3 megasamples per second. This sample rate still gives us plenty of resolution and detail to view the signals with, and over multiple cycles of the signal, it should be relatively easy to get accurate impedance calculations from the system.

After purchasing a demo board for the LTC1407A, I interfaced it with the SPI interface of the PIC32, whose code is located in the Appendix. I did this using framed SPI mode, a built-in feature of the PIC32. This mode of SPI uses a clock to automatically trigger the chip select line at regular intervals. Using 32-bit mode, I could automatically trigger the SPI channel to read as often as it could using my clock speed, and store the two ADC values in the buffer until I read it. Using the built-in UART hardware on the PIC32, I was able to interface the system with PuTTY such that I could see the output of the ADC as it came in. The system can be seen in Fig. 9. This is an extraordinarily simple set up, as the figure shows, only requiring three signals between the two devices. For the purpose of simplicity, two SMA to BNC converters were purchased for ease of integration into the current system, which uses BNC connectors to connect with the PicoScope. With the system on a breadboard, it was difficult to turn up the clock speed very high and test how quickly I could run the ADC. To run the ADC at its highest speed, a PCB would need to be created to minimize the effect of parasitic capacitance.

The demo board, the DC1082-A, only cost \$50, a far cry cheaper than the PicoScope. Upon testing further, I was able to make the system work as expected when testing signals off a power supply on the same ground as the microcontroller and the ADC power supply. However, I soon realized this would not be as easy as I thought for the system at hand without some extra effort. The LTC1407A is a pseudo-differential bipolar ADC, but this means that it can measure the differential analog voltage according to a reference which cannot go below ground level. This problem could be alleviated by using circuitry to bias the incoming voltage and ensure it never goes out of range. Another solution would be to use a pseudo-differential *true* bipolar ADC. These ADCs are able to use the ground pin from the incoming signal and measure how far the signal is from ground with a large voltage swing. For example, the Linear Technology LTC2328-16 is a single-channel ADC, but it can still achieve 1 megasample per second via an SPI channel, and has a ±10.24V dynamic range and 16-bit resolution. With the PIC32MX250F128B, there are two SPI channels, which could be used to roughly parallelize two of these ADCs. This type of ADC would even more closely approximate the behavior of the PicoScope, and a demo board would only be \$100. While these are still not particularly cheap prices, the LTC2328-15 unit itself only costs \$35, and the LTC1407A only costs around \$10. With a custom PCB, these prices could be even cheaper, and neither one comes remotely close to the exorbitant cost of the PicoScope. The code I have written is very general purpose for Microchip PIC platforms, and can easily be used for virtually any SPI-based ADC with some minor tweaks.

3.5. Team Contributions

Throughout the year, I interfaced with the team of Biomedical Engineers, including Emma France, Nariman Ziaee, Robert Pena, Vaishali Bala, and Henry Wang. Originally, we planned to improve all parts of this system, with some working on the probe, some on the vacuum system, and some with the electronics. During this time of the year, I assisted the team in understanding various parts of the system, especially in electronics. I provided assistance to the team in understanding the vacuum system and its control methods with the Arduino at the beginning of the year. As time went on, I also assisted the team in understanding the probe board and main board layouts. The electronic function of all of the components were something of a mystery before this, due to somewhat unclear diagrams. After about 10 hours of dissecting all of the various schematics and documents, the team was able to thoroughly understand the system's function. Additionally, there were multiple brainstorming sessions throughout the first semester about ways to make this system better.

Unfortunately, due to the lack of functionality in both systems, eventually the BME team abandoned the old systems, since it seemed there would not be time to actually improve the system in a meaningful way if we couldn't verify full functionality in the original system. The team began building a new system based on a National Instruments myRIO. I was able to assist in a limited capacity with the development of this system, primarily in the way of debugging the electronics and helping with the new vacuum system development where my expertise was needed.

4. Results

The results of this project are primarily incremental, since there were so many areas to fix and develop. However, each area that I affected during this project will be summarized here.

4.1. Version 1

The work on the V1 system has been primarily in making the system more resilient. One of the first orders of business should be ensuring that the hardware on the V1 device works as expected, such that it can be verified up to the same level as the V2 device has been. Once the probe has been verified, the V1 device can go forward as a development unit such that improvements to the system can go through unit and integration testing before improving the V2 system in a likewise manner. While the V1 device is primarily intended as a development machine, it could certainly use a little bit more system integration. Most of the work this project has accomplished with it involves the connectors, but the wires themselves could certainly be cleaner and better organized.

4.2. Version 2

The V2 device's components that do not involve the probe have been unit tested fully, along with some integration testing. Without a verified probe, however, the entire system cannot be fully verified. The firmware on both microcontrollers is running as expected, and the communications are finally reliable and do not require frequent user attention to ensure the two controllers are connecting to the C# program correctly. Both pressure sensors in the system are verified to be working correctly and feeding back with their promised accuracy range (0-100 kPa range, ± 0.25 kPa accuracy). The V2 device should be modified to either integrate an FT232 USB UART chip to interface with the USB hub, or should change its USB connector to a 3-pin connector that connects to a USB-UART cable in the future, since the current ad-hoc fix using a breakout is not at all optimal.

4.3. Software

The software is finally in a centralized location on GitHub, as well as on the main laptop in lab, with a clean, understandable interface and plenty of comments in the code to explain what is actually going on under the hood. A link to the Github is located in Section 7. I have placed the microcontroller code on the same Github. While neither of the microcontroller codes should need to change in the foreseeable future, it is important that they are on version control should change ever need to occur.

4.4. Data Acquisition Prototype

The data acquisition system prototype will be useful for future students in attempting to make this system smaller and more integrated. The system has plenty of promise for being able to run extremely fast and perform all the necessary calculations. There should be further research into what the best option for the ADC is, since there are multiple tradeoffs to consider. Since using a true bipolar ADC would be simpler, and since the true bipolar ADCs have such a wide voltage range, it could be argued that it is worth the price to be able to purely interface them with a microcontroller and create the fastest system possible. However, the fact remains that if a more limited bipolar ADC like the LTC1407A was used, we still know the general ranges of the voltages we intend to receive, so being able to bias the voltage correctly should not be particularly hard. Either way, the best option forward for the next prototype is a custom PCB that includes a fast microcontroller with two SPI channels so that the system will be subject to as little parasitic capacitance as possible. The faster the microcontroller can run, the more chance the system stands at achieving better throughput from the ADC, giving better results for the system overall.

The code I have written is a proof of concept, and relatively simple in its abilities, but it enables two critical needs for any data acquisition system on this project. The first is fast analog-to-digital conversions and a method for getting these back to the microcontroller with as little latency as possible. The method I currently use is already fast, but if a future version used DMA to accomplish reading in the data, it is possible the system could be even faster. The second need is interfacing with a PC whereby the data can be saved elsewhere and analyzed later. In theory, the device would eventually not need to connect to a PC at all, and should be able to give reliable results without need for extra analysis. However, until then, a UART connection is the best way of enabling all the needs of this project.

The optimal set up for this project would involve integrating the new prototype with the function generator. This would be rather easy, as the SPI interface only requires 3 wires, so a PIC32 could easily interface with both. Additionally, the PIC32 uses DMA to create the function generator capabilities, which do not take any CPU cycles, so data acquisition and analysis on the same chip is certainly an option in the future for the system.

4.5. Team Contributions

The BME team has created a virtually entirely new system that has very little carried over from the old system. This will be very useful in the future in terms of perhaps making the V1 and V2 systems better. For example, the new system includes a small I2C ASIC that can integrate the functionality of the function generator, the PicoScope, and the MATLAB Data Analysis, which would massively shrink the size of the system if integrated into the old systems.

5. Conclusion and Future Work

Overall, this project was an experience in growing complexity. The lack of centralized documentation and version control made understanding this project extremely difficult from the start. Additionally, the variety of systems to understand and the various failure modes that this project experienced were initially very hard to diagnose. The amount of people that have worked on this project combined with the decentralized knowledge made it very difficult to trust any code or design, since errors seemed to be everywhere. However, for the hand this project dealt, remarkable progress was made in giving the two systems a fighting chance for their future.

In the future, this system should focus on becoming even more reliable. A prime example of this is the probe's connection to the flex circuit. There should be some method of being 100% sure that the flex circuit is connected to the probe. Additionally, the BHComBus app on the computer can easily be replaced by an embedded controller and a small screen. This would simplify the system as well as make it a more integrated unit. This is a flaw of both the old systems and the new BME team system, and would be an excellent addition for making this a simple handheld device.

As mentioned before, being able to replace the PicoScope is the most pressing issue for the V1 and V2 system, as it is a massive contribution to the overall price of the device. Being able to implement its functionality with a microcontroller would greatly decrease the cost, bringing it down to 10-20% of its original cost.

6. Acknowledgments

I would primarily like to thank Bruce Land for all of his guidance and assistance through the vast variety of problems I brought to him this semester, including with debugging the system and helping me develop my data acquisition prototype. There were many times where the large scope of this project and all of its problems seemed overwhelming, but Bruce always made the problems seem manageable. I would also like to thank Jack Thompson for helping me in a variety of ways, especially in making V1 a more reliable system. Thanks to Dr. Jonathan Butcher for the financial support in the data acquisition prototype, as well as the opportunity to work in his lab. Thanks to my mother and father, who always cheered me on through all the tough times this year. I also owe the BME team my thanks for accepting me with open arms, helping me out with debugging the systems when I needed help, and always making me laugh.

7. Data Sheets and Schematics

Data Sheets

Git Repository with all code for microcontrollers and PC controller app
(contact `svr24@cornell.edu` for access)
`https://github.com/shivarajagopal/biohaptix/`

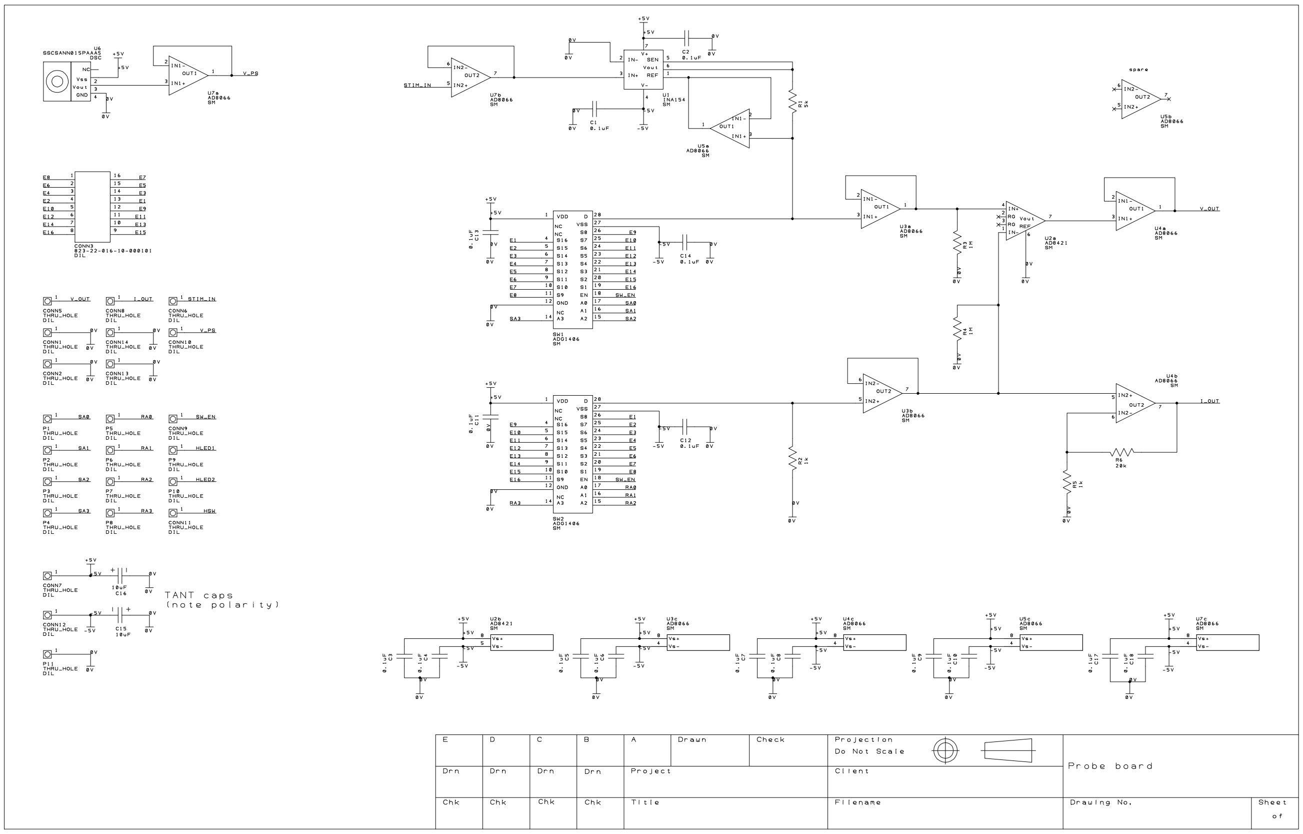
Function Generator Documentation (Mingyuan Huang, Bruce Land):
`https://people.ece.cornell.edu/land/courses/eceprojectsland/STUDENTPROJ/2014to2015/mh2239/index.html`

Linear Technology LTC1407A:
`http://cds.linear.com/docs/en/datasheet/14071fb.pdf`

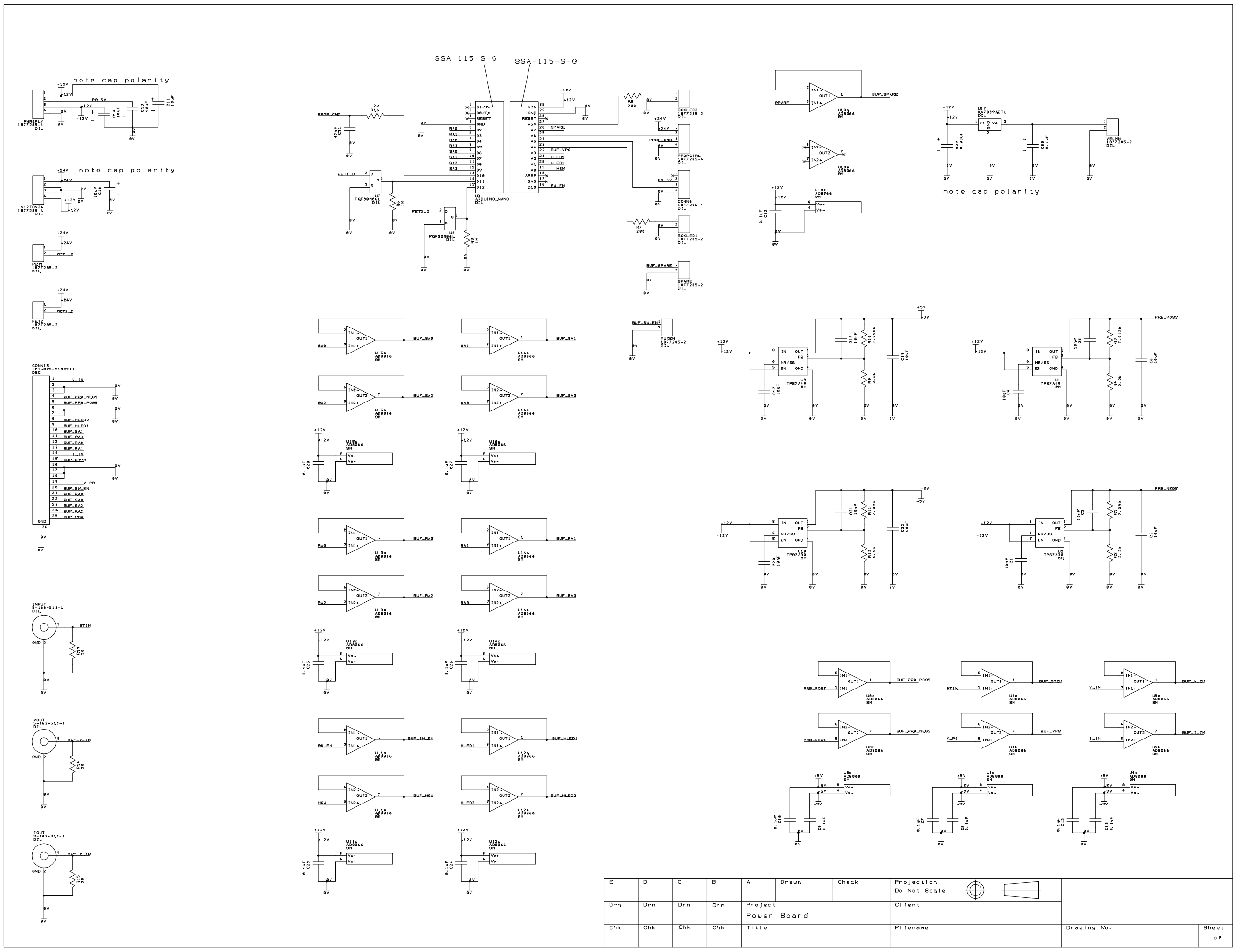
Linear Technology DC1082A:
`http://cds.linear.com/docs/en/demo-board-manual/dc1082af.pdf`

PicoScope 4226:
`http://datasheet.octopart.com/PICOSCOPE-4227-CASE-BUNDLE-Pico-datasheet-8865221.pdf`

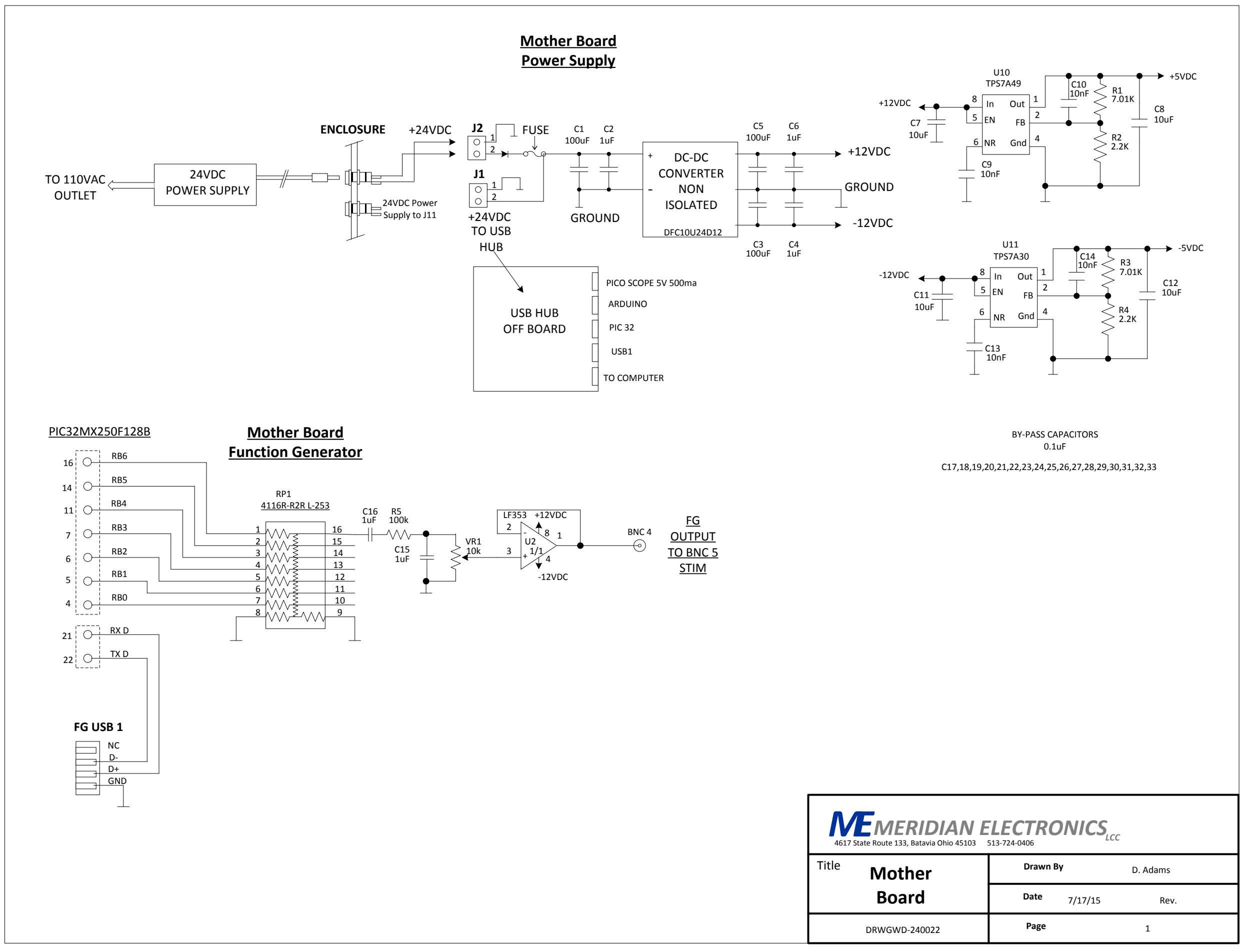
Probe Board Schematic

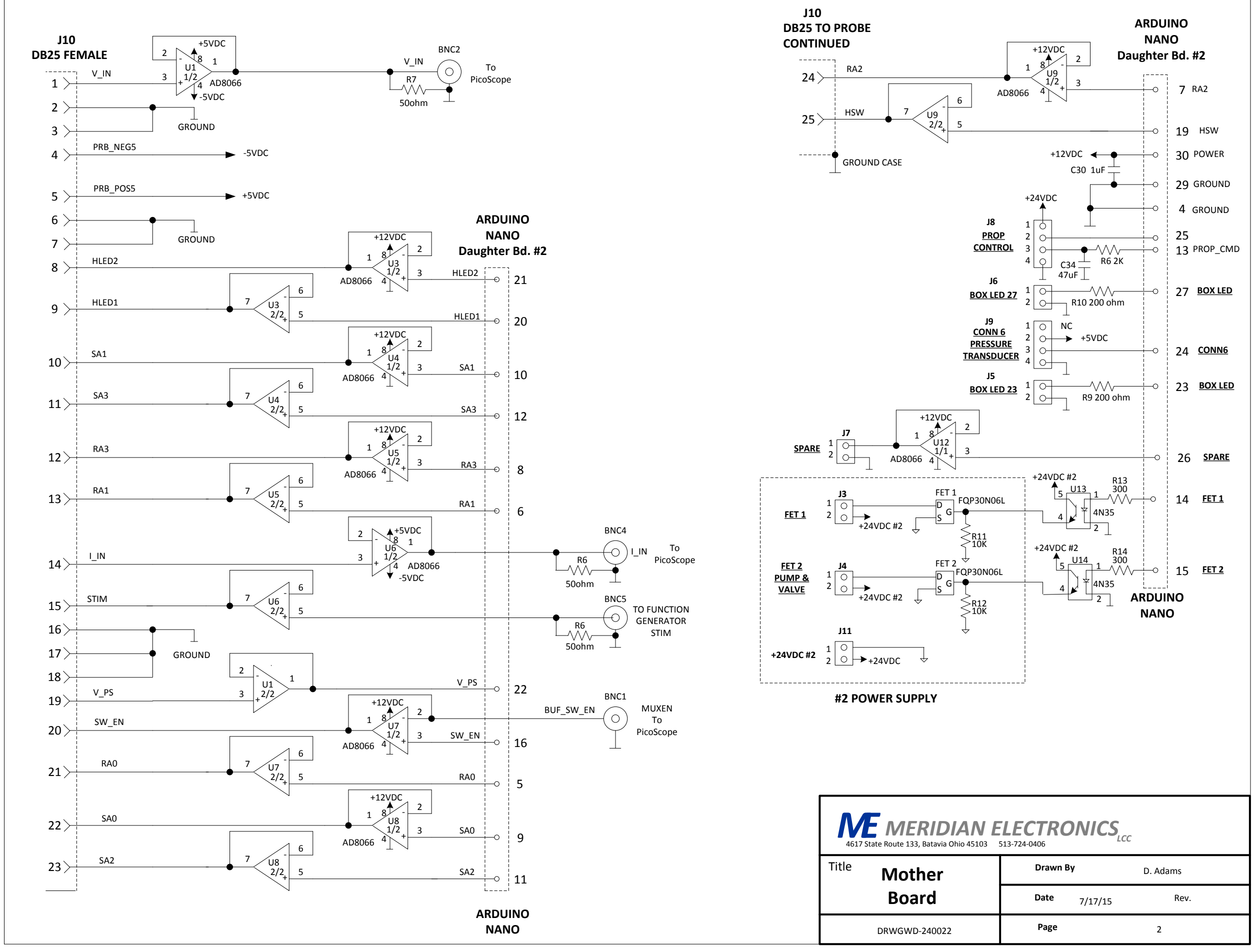


V1 Base Board Schematic



V2 Motherboard Schematic (see errata after figures)





MERIDIAN ELECTRONICS LCC
4617 State Route 133, Batavia Ohio 45103 513-724-0406

Title **Mother Board**	Drawn By	D. Adams
	Date 7/17/15	Rev.
DRWGWD-240022	Page	2

Errata from V2 Motherboard Schematic:

1. In the diagram of the PIC32 (Mother Board Function Generator) on page 21: There should be a cable that includes an FT232 USB UART conversion chip between the microcontroller pins and the USB hub.
2. In the diagram of the PIC32 (Mother Board Function Generator) on page 21: Capacitor C15 should be 100pF, not 1µF.

8. Appendix

`framedspi.c`

The PIC32 code to read the SPI channel and send the value out to a UART channel.

code/framedspi.c

```c
/*
 * File:   framedspi.c
 * Author: Shiva Rajagopal
 *
 * Created on March 6, 2016, 12:34 PM
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "config.h"


#define config1 FRAME_ENABLE_ON | FRAME_SYNC_OUTPUT | SPI_CKE_OFF | SPI_MODE32_ON | CLK_POL_ACTIVE_HIGH | MASTER_ENABLE_ON | SPI_SMP_OFF | PRI_PRESCAL_16_1
#define config2 SPI_ENABLE | SPI_IDLE_CON | SPI_RX_OVFLOW_CLR | FRAME_POL_ACTIVE_HIGH | FRAME_SYNC_EDGE_PRECEDE


// frequency we're running at
#define SYS_FREQ 40000000

// UART parameters
#define BAUDRATE 9600 // must match PC end
#define PB_DIVISOR (1 << OSCCONbits.PBDIV) // read the peripheral bus divider, FPBDIV
#define PB_FREQ SYS_FREQ/PB_DIVISOR // periperhal bus frequency

// useful ASCII/VT100 macros for PuTTY
#define clrscr() printf( "\x1b[2J")
#define home()   printf( "\x1b[H")
```

```
#define pcr()     printf( '\r')
#define crlf      putchar(0x0a); putchar(0x0d);
#define backspace 0x08
#define max_chars 10 // for input buffer

 static char rxchar='0';     //received character
 int num ;
#define str_buffer_size 20
 char str_buffer[str_buffer_size];

// receive function prototype (see below for code)
 int GetDataBuffer(char *buffer, int max_size);


 int spiValBuffer[1000];

 const char *byte_to_binary(int x)
{
    static char b[9];
    b[0] = '\0';

    int z;
    for (z = 128; z > 0; z >>= 1)
    {
        strcat(b, ((x & z) == z) ? "1" : "0");
    }

    return b;
}

/*
 * INIT STUFF
 */
void initialize(void) {
    // Turn off Analog
    ANSELA = 0; ANSELB = 0;

    // UART PPS
    PPSInput (2, U2RX, RPB11); //Assign U2RX to pin RPB11 -- Physical
        pin 22 on 28 PDIP
    PPSOutput(4, RPB10, U2TX); //Assign U2TX to pin RPB10 -- Physical
        pin 21 on 28 PDIP

    UARTConfigure(UART2, UART_ENABLE_PINS_TX_RX_ONLY);
    UARTSetLineControl(UART2, UART_DATA_SIZE_8_BITS | UART_PARITY_NONE
        | UART_STOP_BITS_1);
    UARTSetDataRate(UART2, PB_FREQ, BAUDRATE);
    UARTEnable(UART2, UART_ENABLE_FLAGS(UART_PERIPHERAL | UART_RX |
       UART_TX));

    // SPI PPS
    PPSOutput(1, RPB3, SS1);
    PPSInput (2, SDI1, RPA1);

    // Open SPI and clear buffer
    OpenSPI1(config1, config2);
    WriteSPI1(0);

    rxchar = 0 ; // a received character

    memset(str_buffer, 0, str_buffer_size); // init the string buffer

    clrscr();  //clear PuTTY screen
    home();
}



/*
 *
 */
int main(void) {

    initialize();

    int count, items, num_char;
    count = 0 ; // count the number of characters
    __XC_UART = 2;

    printf("Framed SPI mode\n\r");
    printf("Starting...\n\r");


    int i=0;
    volatile int spiVal;
    volatile int curVal;
    int upperVal, lowerVal;
    while(1) {
        while ( i < 10 ){
            while (SPI1STATbits.SPIBUSY);
            curVal = ReadSPI1();
            spiValBuffer[i++] = curVal;
            WriteSPI1(0);
        }
        for (i = 0; i < 10 ; ++i) {
            upperVal = (spiValBuffer[i] & 0x3FFF0000) >> 16;
            if (upperVal & 0x2000) upperVal |= 0xFFFFC000;

            printf("%d:\t", upperVal);
            printf("%s ", byte_to_binary((upperVal & 0xFF00)>>8));
            printf("%s ", byte_to_binary(upperVal & 0xFF));
            printf("\n");

            lowerVal = (spiValBuffer[i] & 0x3FFF);
            if (lowerVal & 0x2000) lowerVal |= 0xFFFFC000;

            printf("%d:\t", lowerVal);
            printf("%s ", byte_to_binary((lowerVal & 0xFF00)>>8));
            printf("%s ", byte_to_binary(lowerVal & 0xFF));
            printf("\n\n");
        }
        i= 0;
        // wait for a character
        //while(!UARTReceivedDataIsAvailable(UART2)){};
        //UARTGetDataByte(UART2);
    }
    return (EXIT_SUCCESS);
}
```

`config.h`

The configuration file for the PIC32 to properly set up the clock speeds and turn on features

code/config.h

```
#ifndef CONFIG_H
#define CONFIG_H
#define _SUPPRESS_PLIB_WARNING 1
#include <plib.h>
// serial stuff
#include <stdio.h>

//=============================================================
// 40 MHz
#pragma config FNOSC = FRCPLL, POSCMOD = OFF
#pragma config FPLLIDIV = DIV_2, FPLLMUL = MUL_20 //40 MHz
#pragma config FPBDIV = DIV_1, FPLLODIV = DIV_2 // PB 40 MHz
#pragma config FWDTEN = OFF,  FSOSCEN = OFF, JTAGEN = OFF, DEBUG=OFF

//==============================================================
//////////////////////////////////
// set up clock parameters
// system cpu clock
```

```
#define sys_clock 40000000

// sys_clock/FPBDIV
#define pb_clock sys_clock // divide by one in this case

#endif  /* CONFIG_H */
```

References

[1] T. Aoki, T. Ohashi, T. Matsumoto, and M. Sato. The pipette aspiration applied to the local stiffness measurement of soft tissues. *Annals of Biomedical Engineering*, 25(3):581–587, 1997.

[2] A. Gefen and S. S. Margulies. Are in vivo and in situ brain tissues mechanically similar? *Journal of Biomechanics*, 37(9):1339–1352, 2003.

[3] M. Huang. Tissue impedance and biomechanical measurement device modification. 2015.

[4] A. Nava, E. Mazza, F. Kleinermann, N. J. Avis, and J. McClure. *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003: 6th International Conference, Montréal, Canada, November 15-18, 2003. Proceedings*, chapter Determination of the Mechanical Properties of Soft Human Tissues through Aspiration Experiments, pages 222–229. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[5] G. M. Tarsi, R. A. Gould, J. A. Chung, A. Z. Xu, A. Bozkurt, and J. T. Butcher. Method for non-optical quantification of in situ local soft tissue biomechanics. *Journal of Biomechanics*, 46(11):1938–1942, 2013.