# Acoustical Analysis Unit

By

Kristine Cabrera

Kevin Chen

Joseph Shim

Final Report for ECE 445, Senior Design, Spring 2013

TA: Ryan Corey

May 1, 2013

Project No. 14

**Abstract**

Our group designed and built a portable acoustical analysis unit that takes in audio from microphones and displays user chosen analysis, such as a simple FFT or direct audio input waveforms, on a touchscreen LCD. The system has four BNC connections to connect up to four different microphones and contains the actual unit (microcontroller, LCD, ADC, and analog) in one compact device. Each of these various modules will be powered by 8 AA batteries that are easily replaceable when depleted. With this portable unit, researchers, such as Dr. Swenson and Dr. White of the U.S. Army, will be able to comfortably bring this into the field with ease and find/analyze sources of noise.

# Contents

# 1 Introduction

## 1.1 Statement of Purpose

The goal of this project is to design an acoustical analysis unit for Dr. Swenson and Dr. White, that will be used for research. The final deliverable should be able to do real time analysis of acoustical waves between 20Hz to around 10kHz and show the data in a simple and clean manner. Our device takes input from a microphone probe, such as the Brel & Kjr microphones that the army acoustics research lab possesses. The resulting waveforms will be displayed or analyzed by the device, as the user so chooses. The user will be able to treat this like any other lab equipment, as it will have several menus and user inputs for navigation. All the hardware and software will be properly documented so that the research team will be able to add on new functions as needed, making it far better than any consumer off-the-shelf equipment.

## 1.2 Objectives

### 1.2.1 Functions

The unit takes audio inputs from up to four microphones. Each audio inputs data is kept separate from each other. The device outputs acoustical intensity in the form of a graph which is updated in real time on the LCD screen. It will also output the results of any digital signal processing done on the input audio data on the LCD screen in real time.

### 1.2.2 Benefits & Features

The current device used for acoustical analysis with microphone probes is quite cumbersome and heavy to carry around in the field. This is not very suitable since the microphones and device need to be portable in order to be brought around outdoors. Dr. Swenson and Dr. White would like the device to be more user-friendly and more efficient when trying to take in audio data from the microphone probes. Also, they would like the software to be modular such that it can be easier to add different functions.

To solve these problems, our device will be smaller and lighter allowing for use outdoors. It will have an intuitive program that is capable of being easily modified to increase functionality. Finally, the device will also be compatible with the existing 2-microphone probe using BNC connections.

As of right now, the screen on the original device is quite dark. We chose to implement a large 3.5 inch LCD with a backlight. This would give the device a cleaner look without sacrificing any of the data/menus that need to be shown on the screen.

Our device will have an intuitive program and interface so that Dr. Swenson, Dr. White, or other individuals would like to refine the functions already implemented or input additional functions. The user interface should be spread out nicely and not have too many menus. There should be a clear and understandable way of getting the desired functionality.

# 2 Design

## 2.1 Block Diagram Details

Please refer to the block diagrams in Appendix A.

### 2.1.1 Module Descriptions

***Summary***

The acoustic analyzer unit will consist of 3 main parts: power, microcontroller, and display. The overall system will take the audio from the microphones, process it, and show it to the user through the display. The user can make the system display the specific data that he/she wants and in the form he/she wants through manipulation of the user interface.

***Power Module***

This provides different voltages and power to the other modules: Analog circuit, ADC, Microcontroller, and LCD touchscreen.

***Analog***

The audio block consists of the analog circuitry that will take the audio signal from the microphones to the microcontroller. The input will have to be able to take in a frequency range between 20Hz to around 10kHz. In order to do this a chunk of the circuit will be the anti-aliasing filter that will have an upper 3dB rolloff around 11kHz.

The audio block consists of the analog circuitry that will take the audio signal from the microphones to the microcontroller. The circuitry will include any pre-amplification, filtering, and buffering that's needed and will provide a usable signal for the microcontroller's ADC as well as protecting the microphone from any electrical damage and vice versa.

***Microcontroller***

This module controls all the digital components of the system. It receives signal data from the ADC and inputs from the user via touchscreen, processes the data, and then outputs it to the LCD. This module handles the digital signal processing of the incoming audio signal data. It is also responsible for the GUI and handling user inputs. The sub-modules of this module are all parts of the software code.

1. **Data Collection:** The data collection sub-module handles acquiring the audio data from the two external ADC units. Once the sub-module has acquired all 4 sets of audio data (one from each microphone), it passes the data to the DSP sub-module for processing.
2. **Digital Signal Processing (DSP):** The DSP sub-module handles all the processing of the audio data. The sub-module will initially only include the FFT of the audio data as part of the functionality. The user can add their own DSP functions to process the data in this sub-module. Once the data is processed, it will be passed to the display controller sub-module.
3. **User Input** The user input sub-module takes the input signals from the external touchscreen and

manipulates the DSP settings and GUI settings accordingly.

4. **Display Controller** This sub-module will update the LCD with the data obtained from the DSP sub-module. This module is also responsible for the GUI and manipulation of the GUI based on incoming inputs from the user input sub-module.

***Analog-to-Digital Converter (ADC) Module***

The Analog-to-Digital converter units are part of the input system. The ADCs convert the analog voltage signals received into a digital form for use by the microcontroller.

***LCD Touchscreen Module***

This module displays the processed audio data on a LCD screen for the user. It also displays available settings to the user such as displaying the maximum FFT amplitude frequency measurement or ignoring DC amplitude on the FFT graph. The user will interact with the menus displayed on the screen through the touchscreen.

## 2.2 Power

The main source of power is from 8 AA batteries. We need to both lower and higher voltages that the battery provides. Table 1 lists the voltages needed:

**Table 1: Voltage Requirements**

| Module | Required Voltage |
|---|---|
| Analog Circuitry | 2.5V & 5V |
| ADC | 5V |
| Microcontroller | 5V |
| LCD Screen | 5V |

From the battery pack, it needs to be stepped down to 5V using the LM7805 positive voltage regulator. This 5V will be for the ADC as well as the LCD screen. Then, it needs to be stepped down to 2.5 using the LM317 (adjustable voltage regulator). The block diagram of the power module can be seen in the Block Diagrams Appendix Section A page 28.

**Simulations and Calculations**

1. LM317 - Adjustable Voltage Regulator

   According to the data sheet from Texas Instruments for the LM317, the output voltage can be adjustable from 1.25V to 37V. For our purposes, I want to get a voltage of 2.5V.

   The capacitor Ci at the input of the LM317 is not recommended, however, a 0.1$\mu$F capacitor should be sufficient for bypassing. The capacitor C0 is not necessary either, but improves transient response. The value for this capacitor is 1.0$\mu$F. The capacitor CADJ is used for the prevention of a ripple rejection. There is an added 1N4002 diode for protection.

   To get the resistor values for the desired output voltages, we use this equation:

$$V_0 = V_{ref}(1 + \frac{R_2}{R_1}) + I_{adj}R_2$$

   From the data sheet, normally Vref =1.25V and R1=240$\Omega$ . "Iadj" is typically 50 $\mu$A, and can be negligible, but we used it in our calculations.

   For an output voltage of 2.5V, the calculation is as follows:

$$2.5 = 1.25(1 + \frac{R_2}{240}) + 50\mu(R_2)$$

$$\therefore R_2 = 238\Omega$$

   Because we are using standard resistor values ($\pm$5%), we chose to use a 240 $\Omega$ resistor. Below is a snippet from the TI data sheet.

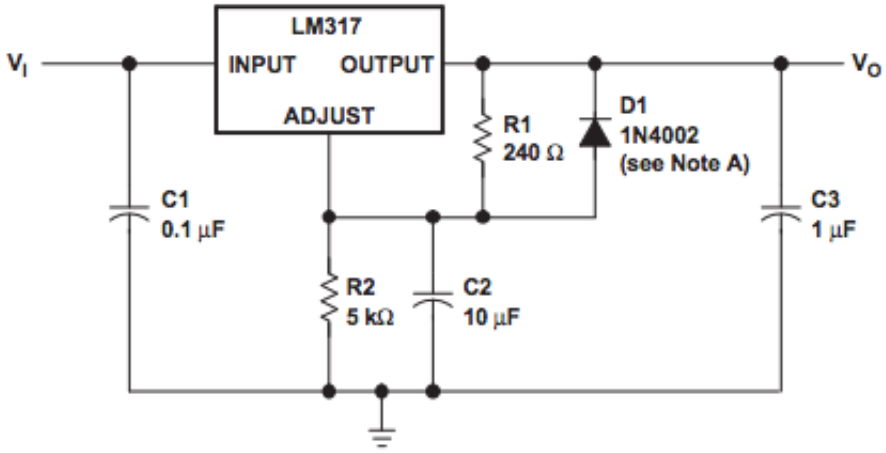


Figure 1: LM317 Typical Configuration [1]

2. LM7805 - Positive Voltage Regulator

   According to the data sheet from Texas Instruments, this chip can have an adjustable output voltage or be at a fixed output of 5V. For our purposes, we chose to keep the it in the fixed output regulator formation. This diagram from TI data sheet is shown below.

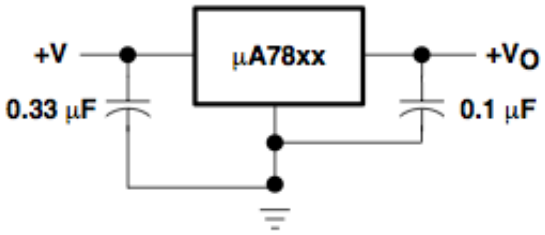


**Figure 1. Fixed-Output Regulator**

Figure 2: LM7805 Typical Configuration [2]

   We only need two capacitors valued at 0.33$\mu$F and 0.1$\mu$F at the input and output of the chip, respectively.

The original design used a 3.7/4.2V LiPo battery. This battery, however, could not supply enough current to the LCD screen and would not turn on. The other modules powered fine; however, we chose to change the battery source to 8 easily replaceable AA batteries instead of the 3.7/4.2 V LiPo Battery. Thus, we removed the charging mechanism (even though this test passed with the LiPo battery). Also, we did not need a Vo=3.3V circuit anymore because we are not using the original clock chip in our first design.

## 2.3 Analog

The Analog Module has to accept frequencies between 20Hz to 10kHz. In order to do so, we used a low pass filter with a decoupling cap. When designing the filter, we need to make sure that the dB output is stable within our operating frequencies as well as having a linear phase change.

In the end, we decided to go with a second order filter to have a steeper drop-off. Since we are working with a small frequency range, we feel that we need at least a second order filter to match our needs. Figure 3 shows the filter design.

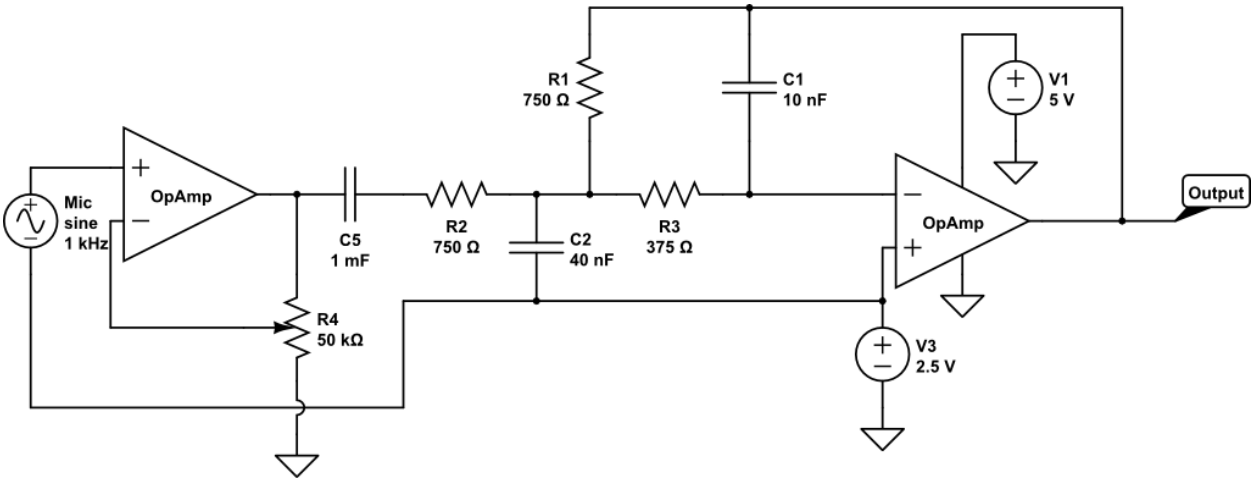


Figure 3: Input Circuit Sim Schematic

Even though the highest frequency we will be working with is 10kHz, we did not choose this as our upper cutoff. If 10kHz were selected as the upper cutoff, then at 10kHz the dB would have dropped around -3dB, theoretically. Knowing this problem can occur; we decided to choose a frequency that is higher; in our case it is 15kHz.

The following is the equation we have to solve to get the proper component values.

$$F_0 = \frac{1}{2*\sqrt{2*\pi*R*C}}$$

Even though we are solving for the product of R and C it was in our best interest to pre-select the values for the capacitor. It would be significantly harder to mix and match capacitor values to come up with the correct value. Resistors on the other hand have more flexibility and easier to work with. We chose to use a .01uF capacitor which is a standard and abundant value.

$$15kHz = \frac{1}{2*\sqrt{2*\pi*R*.01\mu F}}$$

The selection of our OpAmp was determined by several factors such as the amount of noise, bandwidth, supply voltages, settling time, and also based on the applications notes of the datasheets. The OpAmp that was chosen is the OPA227PA that has the following specifications

- low noise @ 3nV/Hz
- wide bandwidth @ 8MHz
- settling time @ 5us
- open loop gain @ 160dB
- input bias current @ 10nA
- offset voltage max @ 75uV
- supply voltage @ ±2.5V to ±18V

With the following information above, we were able to run a simulation to get the results shown in figure 11.

We can see that the frequency response if pretty flat within our operating frequency range. Our phase plot is also pretty stable until we get to the much higher frequencies.

**Pressure Microphone**

The pressure microphone that will be used with the project has a sensitivity of 3.1 mV/p. Keeping this rating in mind, we can calculate the highest expected output for the microphone which in turn can tell us if any special protection circuit is required.

Referencing a dB chart with a wide range of values, the highest noise recorded was a firework explosion from 3 feet away. This caused a dB rating of 150dB, which is louder than standing beside a jet engine. When this dB rating is converted into Pascals, we get 632.456p. Using the sensitivity, we can get the expected voltage output, which is calculated below:

632.456p * 3.16mV/p = 1998.56mV

With an expected max output voltage of 2V, we know that this is well within our projects safe operating parameters.

**Amplifier**

After getting our hands on the actual microphone from CERL we noticed that the micrphone output signal is really small for normal audio ranges (Office enviroment to people yelling). While speaking with a stern tone and citing the alphabet we managed to get around 15mVpp from the microphone. This cause some problems mainly due to the fact that the enviromental noise seemed to be between 5mVpp - 10mVpp range in the UIUC fab lab and around 7mVpp - 13mVpp in the senior design lab. We realized that we need to increase the signal to noise ratio and the solution we thought of to do this was have an amplifier before the filter stage of the analog circuit. The amplifier is just a traditional active noninverting amplifier and can be seen in figure 3 near the input.

## 2.4 ADC

For the ADC Module we wanted to use a 16 bit ADC. After a bit of researching we have settled on the PCM 1801 ADC which was chosen for its apparent simplicity of use, cost, and popularity of the PCM family by

Texas Instruments. The circuit for the ADC chip was designed by looking at the datasheet parameters as well as some input from the partial example given in the datasheet.

This entity originally consisted of a PCM1801 chip. The chip operates in I2S slave mode. The FMT pin on the chip needs to be set high for I2S format data. The PCM1801 digital communication uses I2S communication protocol in slave mode. This particular chip requires three clock signals, a system clock (SCKI), a bit clock (BCK), and a left right channel clock (LRCK) [6]. Originally, our design used an external crystal oscillator and a PLL chip to provide the system clock with the microcontroller providing the bit clock and the left right channel clock.

Unfortunately, the PCM1801 refused to output data during testing. Later on we determined the problem lied with the clocks which apparently needed to be synchronized. We had 2 contingency plans to fix this problem. One solution was to have the microcontroller generate all the clock signals including the system clock. The theory being that then all clocks will be synchronized since they are all divided off the microcontroller system clock. Unfortunately this setup did not generate data out of the ADC chip. The 2nd fix was to have the clocks divided off the PLL system clock in hardware. The bit clock was divided to 2MHz and the left right clock derived from the bit clock divided by 16 cycles. Testing this setup, we saw the ADC chip produce bits of data. However, the microcontrollers communication setup refused to accept these bits. This will be elaborated on in the design of the data collection sub-module of the microcontroller. Due to both plans failing, we decided to fall back to using the on-board 12-bit ADC on the microcontroller. Since the 12-bit ADC operating on a 0-to-+3.3V voltage range instead of the original 5V range, the analog circuit required some adjustments from this change.

## 2.5 Microcontroller

The microcontroller we chose to use was the TI Stellaris Launchpad featuring a Stellaris LM4F120H5QR chip [4]. This microcontroller has an ARM Cortex M4F core which features a floating point unit [7]. We chose this microcontroller mainly for the floating point unit so that any DSP application would be processed efficiently. The main function in the code of the microcontroller initializes the different sub-modules and then runs an infinite loop that continually checks for a new data flag and processes messages that have queued from the touchscreen and GUI. The general flowchart for the code is shown in figure 4 below.

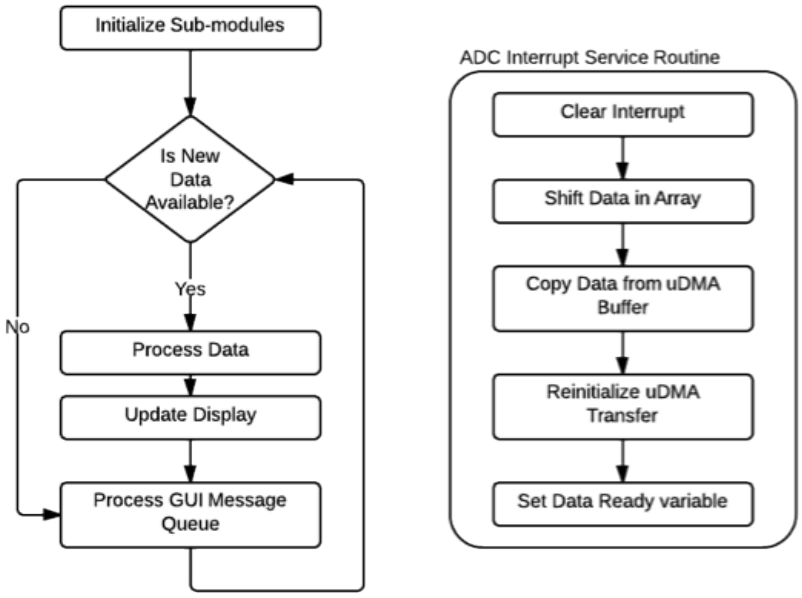


Figure 4: General Software Flowchart

1. ***Data Collection***

   Initially, this sub-module was designed to use the internal SSI peripheral to communicate with the ADC chip. This peripheral is set to Master mode with Freescale SPI Frame Format with clock polarity bit set to 1 and the phase control bit set to 0. It was also set up with DMA to automatically initiate SSI data transfer continuously. The sub-module will first initiate a dummy transmit to start the bit clock which was set to 2MHz and then use the same clock receive data. During every interrupt after data is received by the SSI peripheral, a GPIO pin is toggled to be the left right clock. The digital circuit for this design is shown in figure 5 below. Unfortunately, the ADC chip did not generate data with this setup.

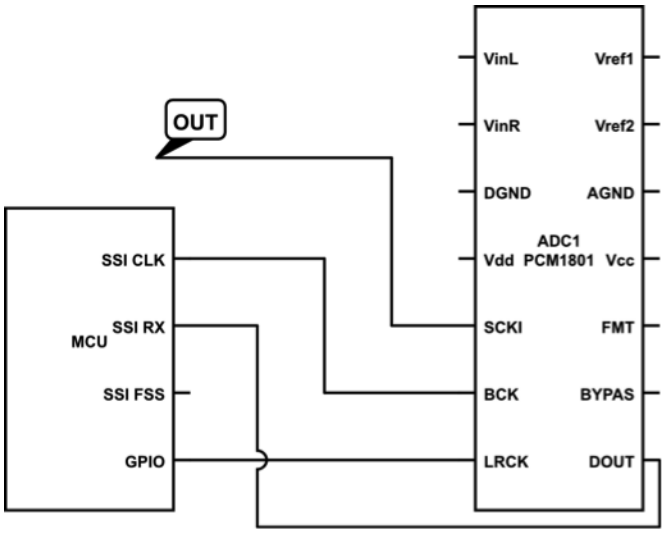


Figure 5: Initial I2S to SSI Communication

Due to the ADC failing to generate data, we had two contingency plans. Our first plan was to generate the clocks directly from the microcontroller. The microcontroller generation of the clocks used PWM to generate a system clock for the ADC chip. This involved dividing the microcontrollers internal clock running at 80MHz by 5 to output a 16MHz clock. The SSI clock which provides the bit clock is still set to 2MHz. All SSI setup is still the same as the initial design. Also the FSS signal of the SSI protocol is fed back into the microcontroller through a GPIO pin. This GPIO pin is set to interrupt on detecting a rising edge. During this interrupt, the program will toggle a GPIO pin to be the left right clock. Since FSS pulses before each SSI transfer, this creates a great way of toggling left right clock to fit the clock pulses. The digital circuit for this setup is shown in 6 below. The testing of this setup is in the Testing section of this report.

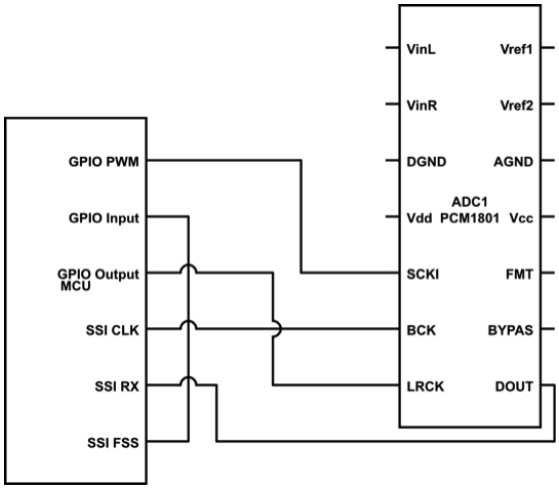


Figure 6: Microcontroller Clock Generation Digital Circuit

Our third design involved a hardware clock divider circuit providing the clock signals. On the microcontroller, the SSI is therefore set to slave mode since it is not generating the clock signals. The left right clock is then passed into the FSS input of the SSI peripheral to initiate SSI transfer on a rising edge. Since the left right clock is actually only toggled high for just the left channel, the left right clock is also passed through a not gate and then fed into a 2nd SSI FSS input. This way the 2nd SSI peripheral will trigger on the right channel data transfer while the 1st SSI peripheral will trigger on the left channel data transfer. All other settings are set to the same. The digital circuit is shown in figure 7. This design is based off TI appnote SPMA042A [10]. The testing of this setup is in the testing section of this report.

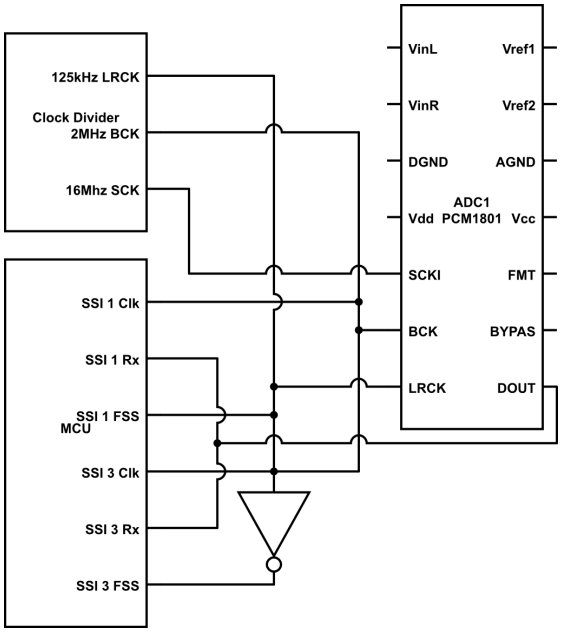


Figure 7: Hardware Clock Generation Digital Circuit

Since none of the fixes we implemented ultimately worked, we fell back to our backup plan of using the on-board 12-bit ADC. This 12-bit ADC was setup to be triggered by an on-board timer which was set to a user-specified sampling frequency. The ADC, using DMA, collects 1024 samples in hardware across 4 input lines (256 samples per line) before interrupting the microcontroller. During every ADC interrupt trigger, the data structure holding the past samples of the ADC shifts the samples over by 256 samples and then inserts the new data onto the end of the data structure. The code then sets a variable to acknowledge the existence of new data to be processed. The final pinout for this is shown in the figure 32 found in Appendix C.3.

2. ***Digital Signal Processing***

The DSP sub-module of the microcontroller uses the ARM CMSIS DSP library for the FFT function. TI appnote SPMA041B details how to build the CMSIS library for use with Stellaris microcontrollers in Code Composer Studio [11]. The DSP sub-module generates a 1024 sample Hamming window if the user chooses to apply it to the audio data before the FFT. The FFT used is a complex FFT of size 1024. The audio data is used as the real parts of the signal and zeros are inserted as the imaginary parts of the signal.

3. ***User Input***

   The user input sub-module uses the touchscreen driver provided with the touchscreen and the Stellaris Launchpad. It also consists of the widgets included in the graphical library included in StellarisWare [9]. Some modification of the touchscreen driver had to be done since it also consists of using one of the on-board ADCs triggered directly by a timer. Unfortunately, Stellaris microcontrollers only allow for one ADC to be triggered by one timer. So instead, we changed the driver so the timer causes an interrupt and then the ISR of that timer directly processor triggers the ADC for the touchscreen. On the software end of things, the user input sub-module consists of many functions that are called when certain types of widgets (such as pushbuttons, checkboxes, or listboxes) are detected to have been touched. These functions check which widget has been touched and execute code or change parameters accordingly.

4. ***Display Controller*** The display controller sub-module uses the LCD screen driver provided with the touchscreen and the Stellaris Launchpad. It also makes extensive use of the graphical library included with StellarisWare [9]. The different widgets are arranged in a tree structure with the root of the tree structure being a canvas widget representing the different pages of the screen. On the audio input and FFT page, the graphs are drawn on the first load of the page. The graph drawing has a separate helper function so that the graph can be redrawn whenever the new data flag is raised without redrawing the entire page. The setting pages are giant tree structures of widget objects that enable different settings. Each page has a set of pushbutton widgets to access other pages. The different pages are shown in figure 8


Figure 8: GUI Pages

### 2.6 LCD

The LCD Touchscreen is a Kentec 3.5 320x240 LCD touchscreen [5]. It is a boosterpack designed specifically to be used with the Stellaris Launchpad. There is driver code provided for use with the touchscreen available. This code makes the LCD screen compatible with the graphical library provided in StellarisWare. The LCD screen communicates with the microcontroller through an 8-bit parallel data line and the touchscreen uses a 4-line analog line through the microcontrollers internal ADC [5].

## 3 Requirements and Verifications

Please refer to the requirements and verifications tables are found in the Appendix Section B.

### 3.1 Power

**Requirements Summar:** This should provide enough/stable power and voltages to each of the other modules that require power.

**Verifications Summary:** Use multimeters to probe the circuit to see if the correct voltage is created for the different modules. We will also check that voltage/power over time for each module for stability purposes.

### 3.2 Analog

**Requirements Summary:** Needs to take in the correct data from the microphone probes and send this to the microcontroller unit.

**Verifications Summary:** Use an oscilloscope to takes measurements from the audio module. We will also run stress tests to see if voltages and current are within the tolerances of key components.

### 3.3 Microcontroller Unit

**Requirements Summary:**This module should receive data from the ADCs without error, process the data, and then output it to the LCD module without error. It also needs to be able to accept user input and act accordingly depending on what pushbutton is pressed and what is being displayed on the LCD currently.

**Verifications Summary:**We will check for valid connections from each of the various modules to the microcontroller. Afterwards, we will use a simple program known to be bug free with audio input and check for expected outcome. Further tests will include ability to take inputs from the User Interface as well as properly displaying text from the MCU.

### 3.4 ADC Module

**Requirements Summary:** This module needs to successfully digitize the input audio data and send it to the microcontroller without errors in I2S format.

**Verifications Summary:** Given a certain input to the ADC, we can take the data received by the microcontroller and check if the data received matches the expected output for the given input.

### 3.5 LCD

**Requirements Summary:** The LCD screen needs to clearly show the menus and data from audio. It needs to be able to show fluid changes of text/graphs on screen.

**Verifications Summary:** The LCD module can be verified using the graphical library example program made by Stellaris to showcase the graphical library. If the example program runs and the expected graphical elements are shown on screen, then the LCD Module works.

## 4 Testing

#### 4.0.1 Power

One of the requirements for power is that the voltage ripple would be within 1V. Figures 9 and 10 are some of the waveforms to calculate the voltage ripple of the two voltage output circuits.

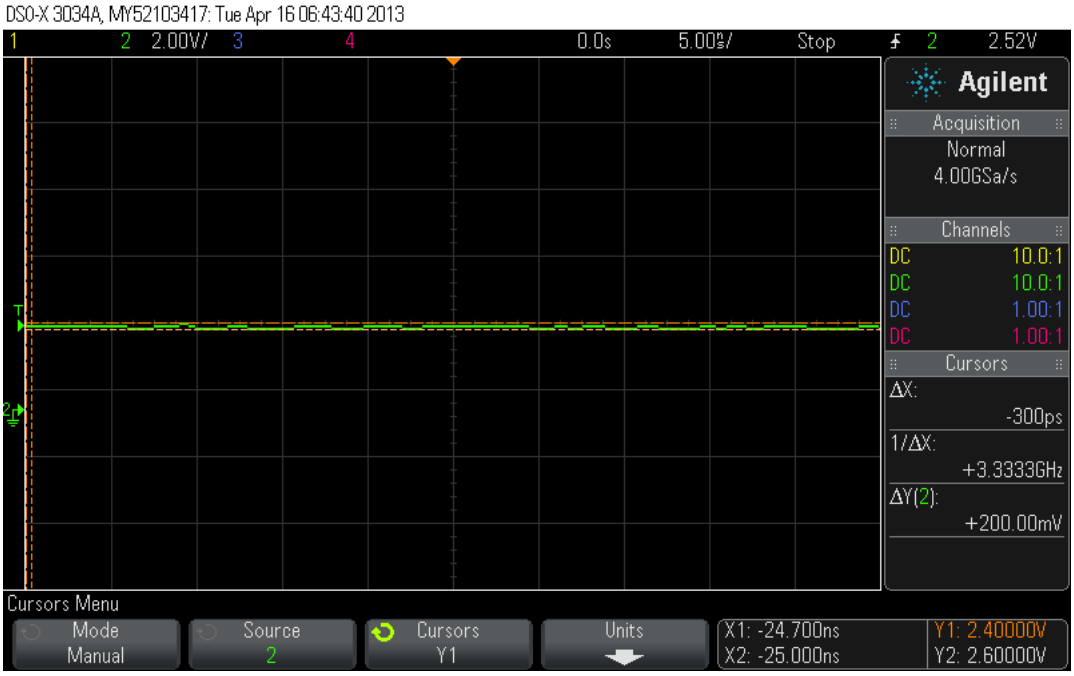


Figure 9: Voltage Ripple for Vo = 2.5 V

From figure 9, we can see that $\triangle$Y = 200 mV, which is less than the max voltage ripple in the requirement.

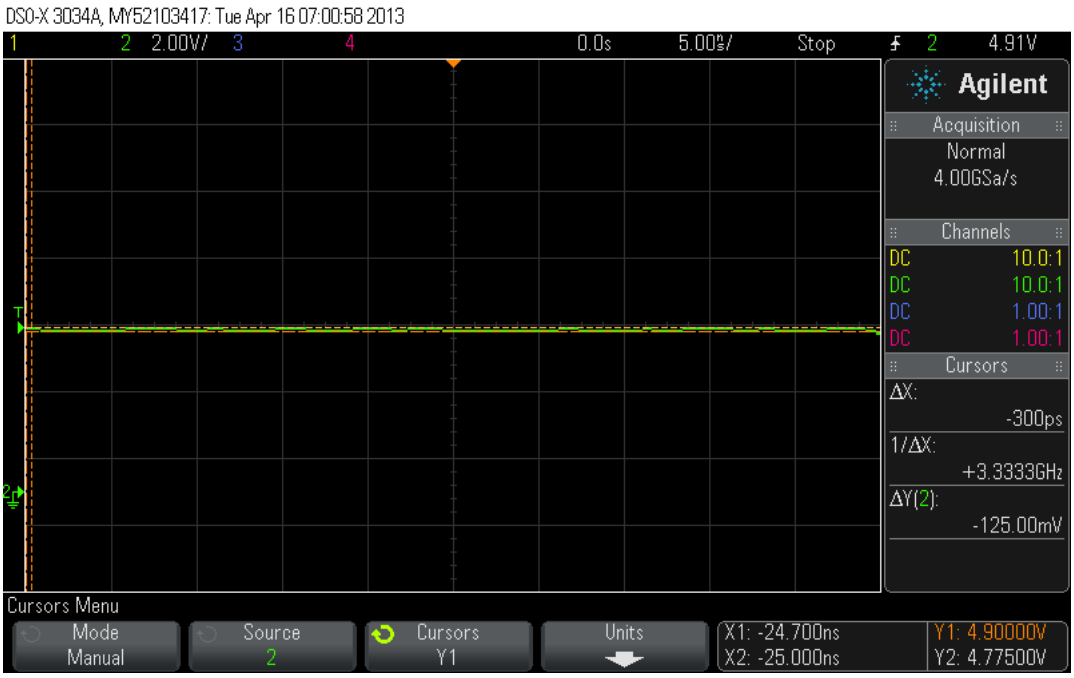


Figure 10: Voltage Ripple for Vo = 5.0 V

From figure 10, we can see that $\triangle$Y = 125 mV, which is less than the max voltage ripple in the requirement.

The power supply is stabilized and the AA batteries are sufficient.

### 4.0.2 Analog

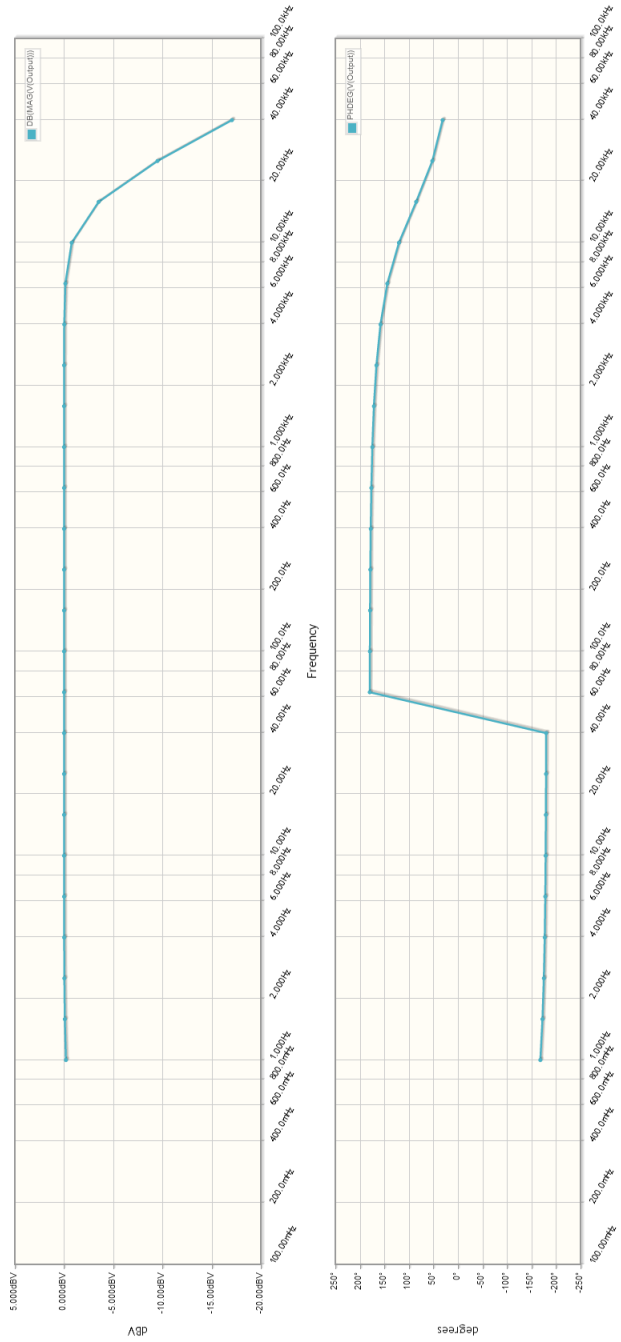


Figure 11: Analog Simulation Results

- Initial Test Results I was responsible for testing the analog and part of the ADC module. The analog circuit was tested on a breadboard and I am currently waiting for pin converters to come in (Should be here March 14th 2013). The parts for the pin converter was ordered more than a week in advance however I have been recently notified that they have shipped. I would have soldered some wires to the ADC (Which is a SMD Component) however there were no soldering iron fine enough to heat up each

pin individually.

Shown Below are the figures from the first verification step. Figures 12 - 15 shows us that in the desired operating range the output matches the input Vpp and frequency. There is some descrepency as later on I realized that the function generator is not completely accurate and is outputing a voltage higher that it shows but further testing shows that the voltage shown in figures 12 - 15 matches the Vpp output from the function generator. In figure 16 you start to see the output amplitude drop off around 15KHz and significantly by 20KHz as seen in figure 17. Figure 18 is the second verification test where I put in a 5KHz sin wave to see if it matches the input amplitude. It does.

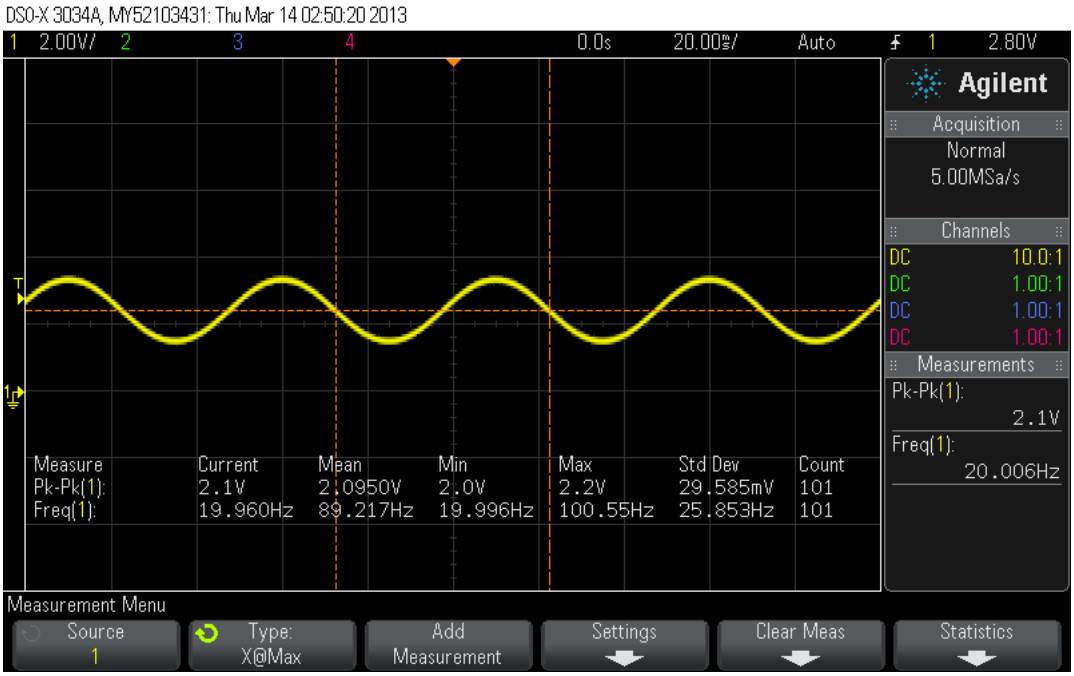


Figure 12: 2Vpp AC Sin Wave @ 20Hz

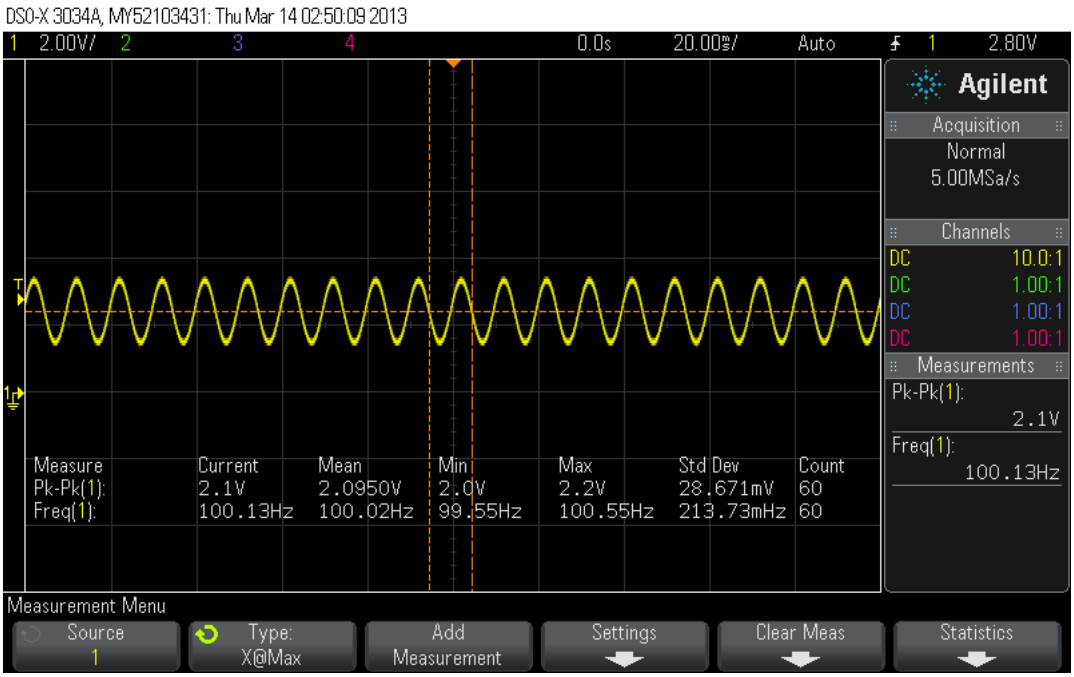


Figure 13: 2Vpp AC Sin Wave @ 100Hz

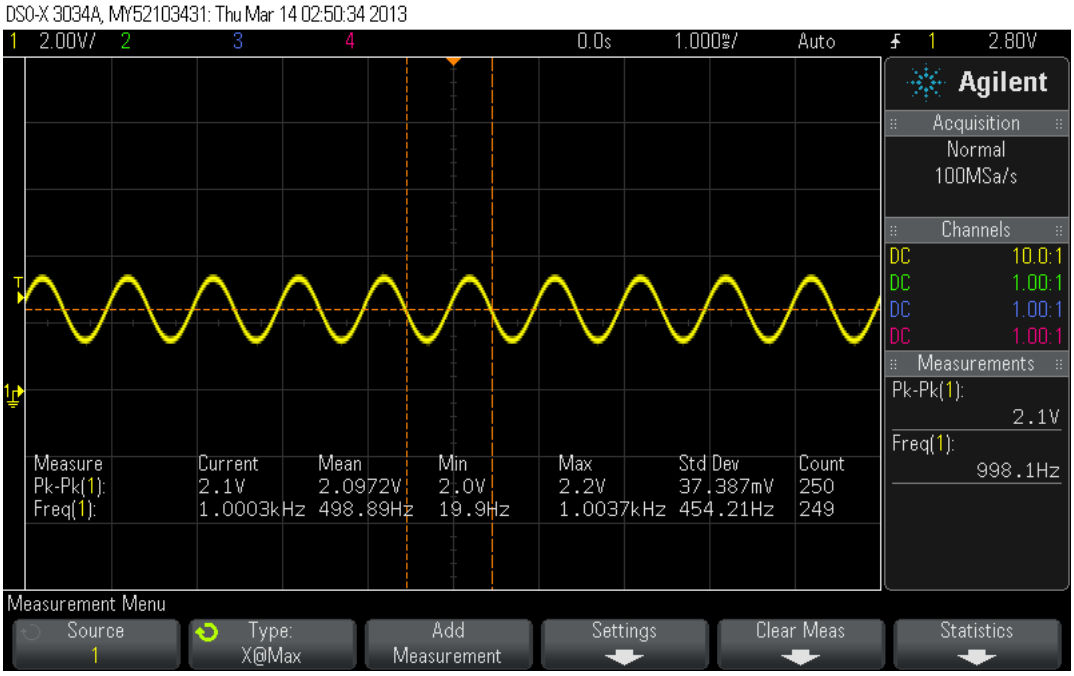


Figure 14: 2Vpp AC Sin Wave @ 1KHz

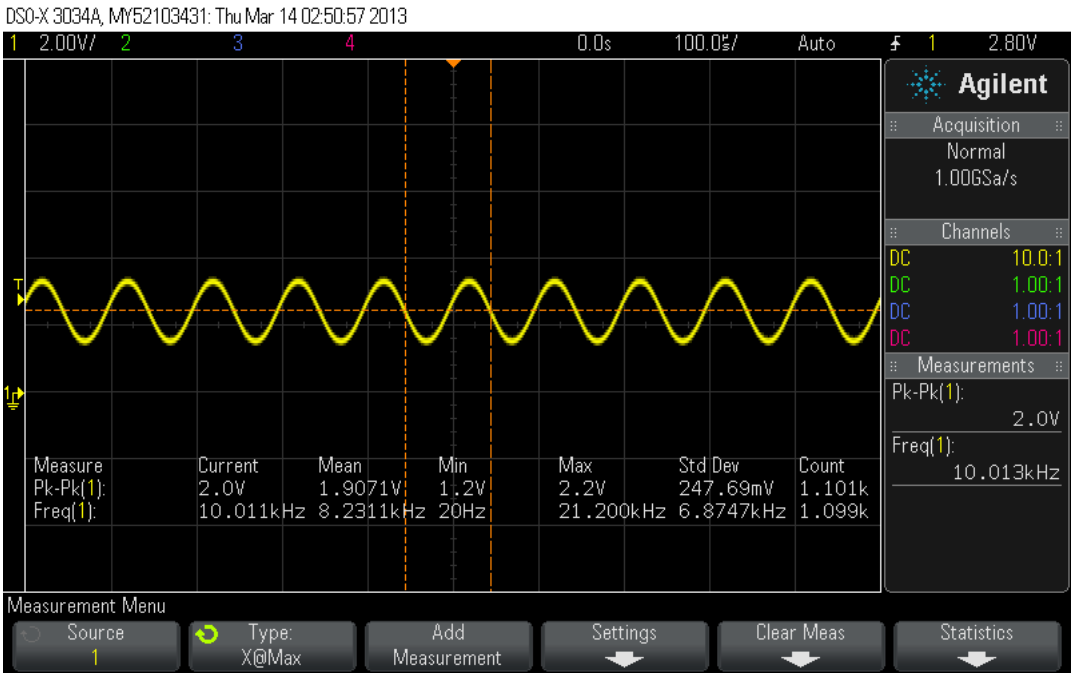


Figure 15: 2Vpp AC Sin Wave @ 10KHz

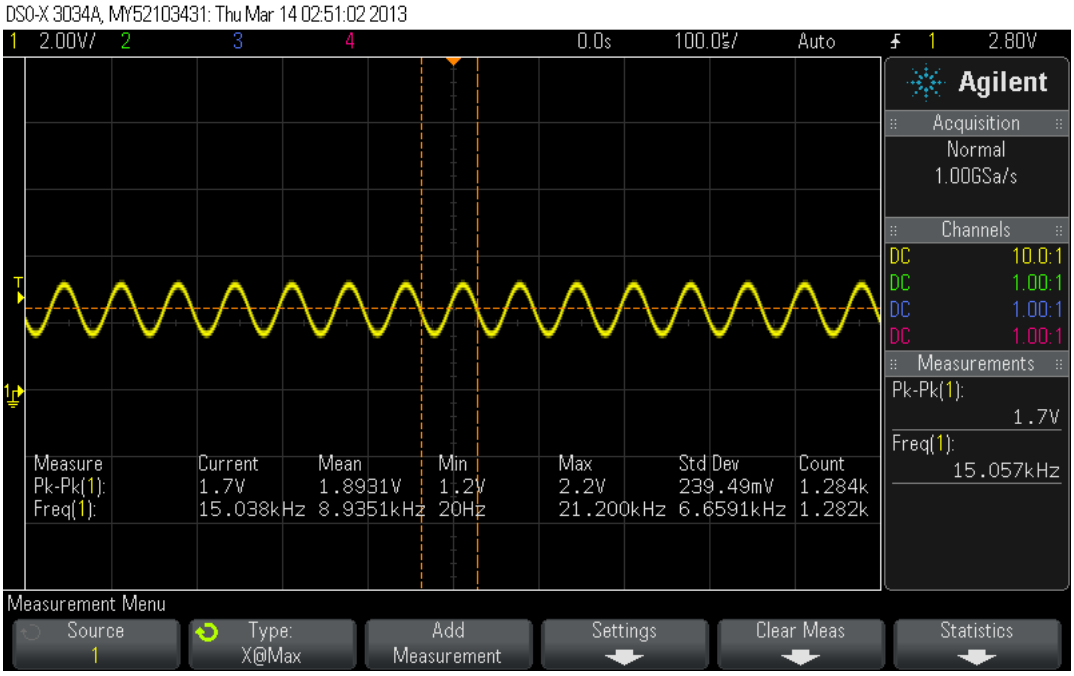


Figure 16: 2Vpp AC Sin Wave @ 15KHz

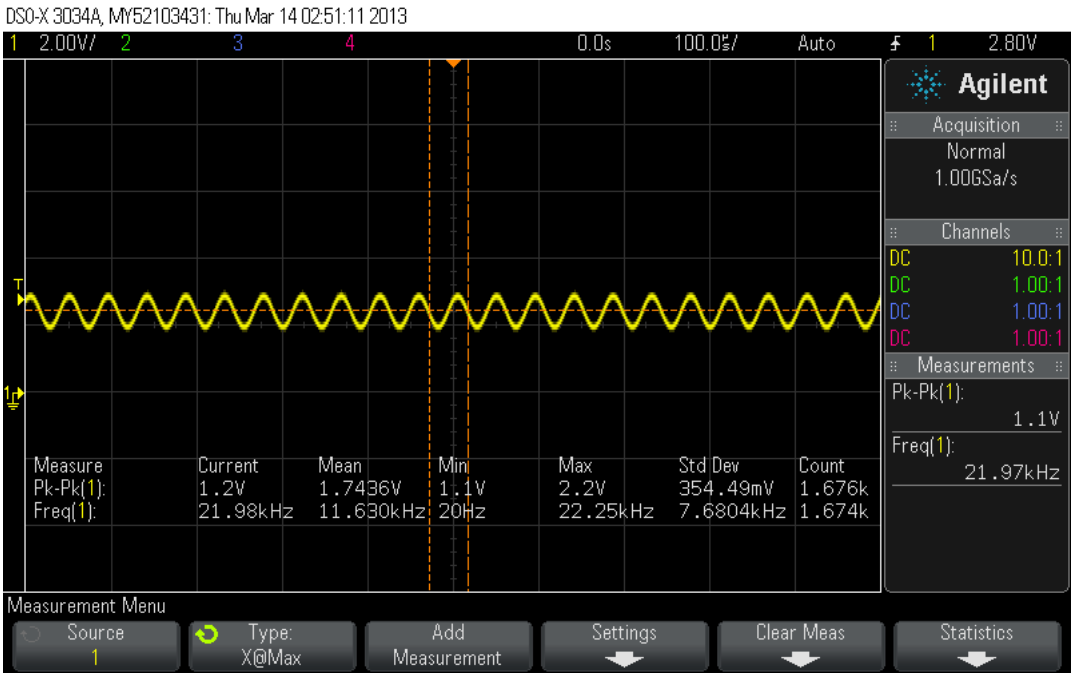


Figure 17: 2Vpp AC Sin Wave @ 20KHz

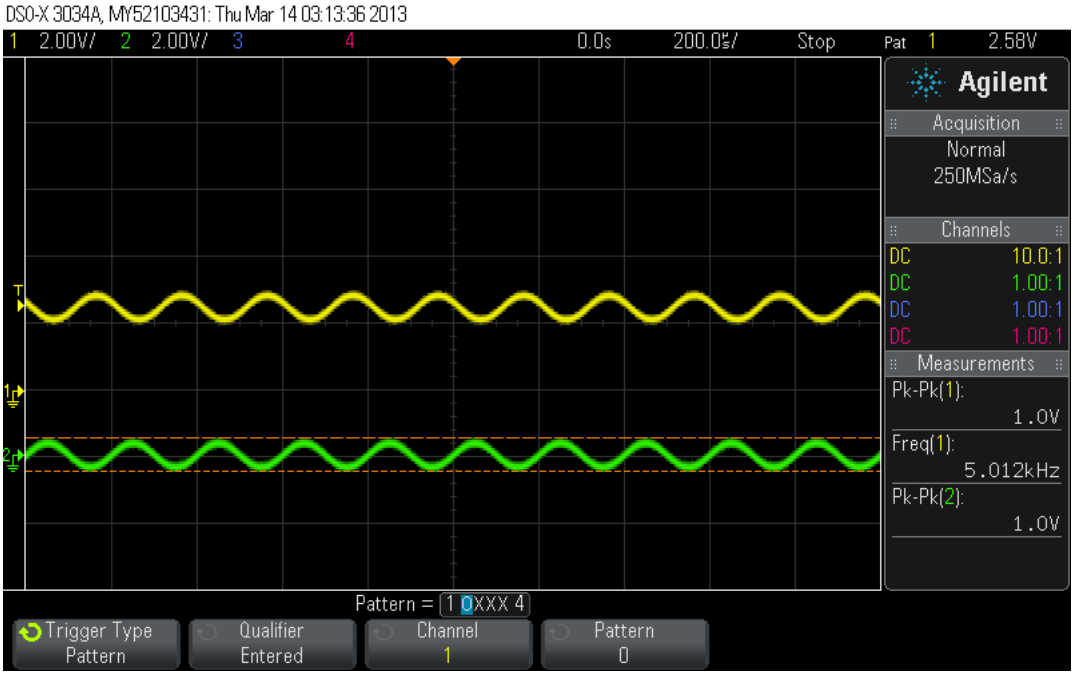


Figure 18: 2Vpp AC Sin Wave @ 20KHz

### 4.0.3 Microcontroller

**Data Collection**

Testing the data collection sub-module involved testing the different plans to generate clock signals. The resulting signals from software generation of clocks are shown in figure 19. As is shown, the signals are very clean and look just like the I2S signals described in the ADC data sheet in figure 20. However the ADC still did not output data as shown on the logic analyzer in figure 19.

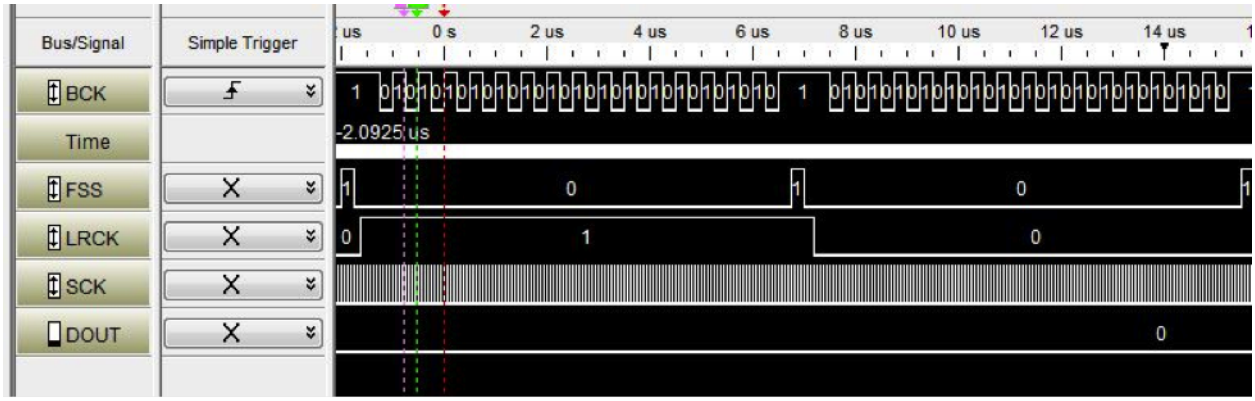


Figure 19: Microcontroller Clock Generation Signals

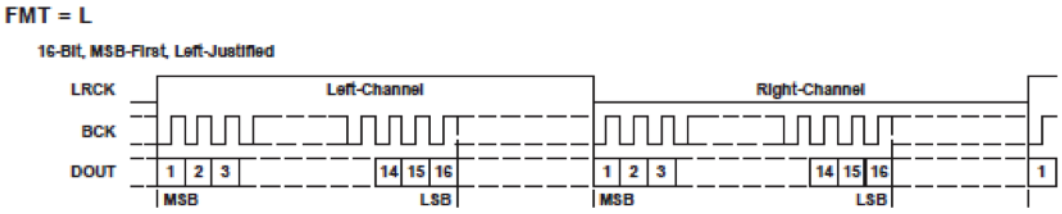


Figure 20: I2S Left-Justified Communication Protocol [6]

Next is the testing of the hardware generated clocks. As can be seen in figure 21, the ADC chip does generate data from the hardware clock signals. However the SSI peripheral refuses to receive data based off these clocks possibly due to lack of a delay between SSI clock bursts as is pictured in figure 19 when the microcontroller generates SSI signals and also in figure 22 which is the SSI protocol signal diagram from the microcontroller datasheet.

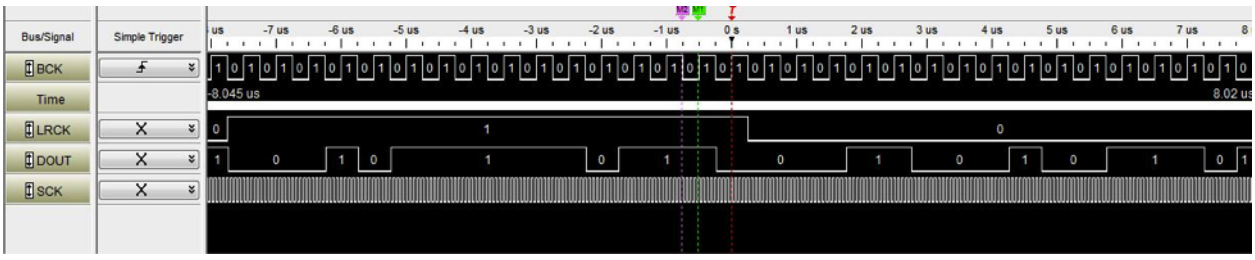


Figure 21: Hardware Clock Generation Signals

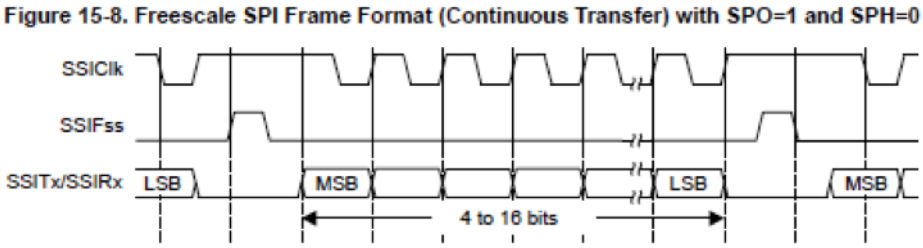


Figure 22: SSI Signal Timing [7]

### Digital Signal Processing

The DSP sub-module was tested by generating a discrete sine wave and applying the FFT to said sine wave. An IFFT was also applied to the FFT data and checked for the original waveform to verify the DSP functionality. The waveforms are shown in figures 23 and 24. These waveforms are plotted directly from the microcontrollers memory. Ignore the apparent frequency modulation in the generated sine wave. That is from the interweaving of zeros as the imaginary part of the data.

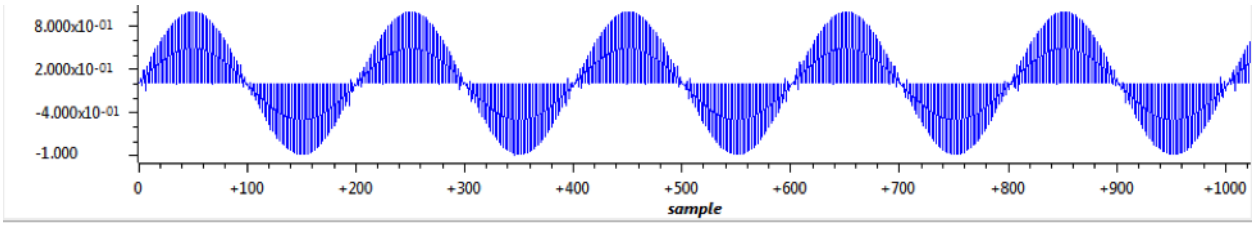


Figure 23: Generated Sine Wave

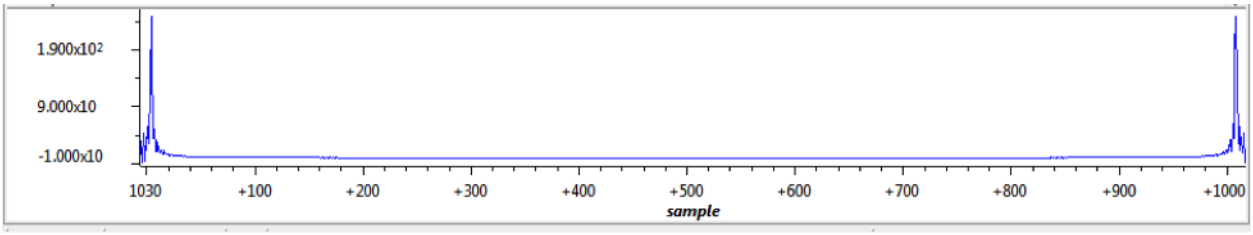


Figure 24: Resulting FFT

### Display Controller and User Input

Both the display controller and user input sub-modules were tested by a program we wrote to display both the generated sine wave and resulting FFT from the DSP testing onto the screen with pushbutton widgets to change between the pages that displayed the waveforms and a settings page that allowed changing the frequency of the sine wave. The visual results of this program are shown in figures 25 and 26. The pushbuttons work when touched to change between the different pages.

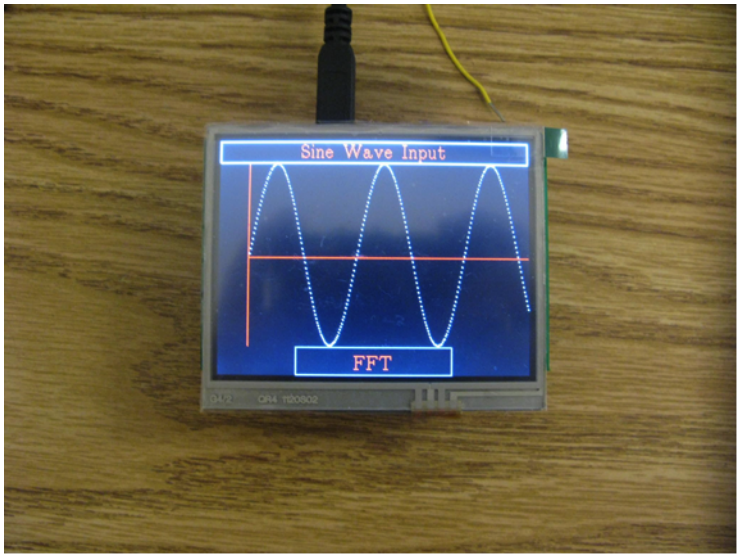


Figure 25: Sine Wave Display

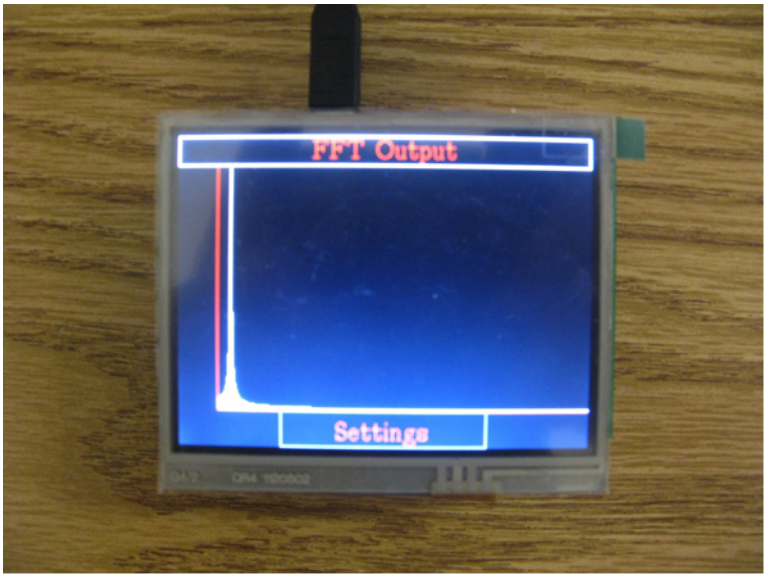


Figure 26: FFT Display

#### 4.0.4 LCD

Please see the display controller and user input testing section above.

# 5 Costs

## 5.1 Parts and Equipment

**Table 2: Parts**

| Parts | Quantity | Cost per Unit | Total |
|---|---|---|---|
| Stellaris LM4F120 | 1 | $14.29 | $14.29 |
| Kentec LCD Boosterpack | 1 | $35.00 | $35.00 |
| USB Cable - Mini | 1 | $4.00 | $4.00 |
| Voltage Regulator - LM317 (SMD) | 1 | $1.06 | $1.06 |
| Voltage Regulator - LM7805 | 1 | $0.69 | $0.69 |
| OPA227PA Op-Amp (SMD) | 4 | $4.40 | $17.60 |
| PCB | 1 | $9.84 | $9.84 |
| AA Battery - 4 Pack | 2 | $3.77 | $7.54 |
| Quad AA Battery Holder | 1 | $1.28 | $1.28 |
| BNC Connectors | 4 | $5.57 | $22.28 |
| SMD Capacitors | 31 | $0.60 | $18.60 |
| SMD Resistors | 18 | $0.65 | $11.70 |
| Potentiometer | 4 | $1.22 | $4.88 |
| **Total** | | | **$148.76** |

**Other Equipment Used**

- 2-Microphone Probe - Property of CERL
- Computer for coding/ programming
- Senior design lab equipment

### 5.2 Labor Costs

**Table 3: Labor**

| Member | Hourly rate | Total hours | Subtotal x 2.5 |
|---|---|---|---|
| Kristine Cabrera | $45 | 140 | $15750 |
| Kevin Chen | $45 | 140 | $15750 |
| Joseph Shim | $45 | 140 | $15750 |
| **Total** | | | **$47240** |

### 5.3 Grand Total

**Table 4: Grand Total**

| Labor | Parts | Total |
|---|---|---|
| $47240 | $148.76 | **$47388.76** |

## 6 Conclusions

### 6.1 Accomplishments

In the end we had a successful, working deliverable. We fulfilled the overall goal - to take inputs from microphones and display them or calculate an FFT on the LCD. The three of us worked together well and were well paced throughout the semester. When we ran into trouble we were quick to combat these problems and ultimately delivered a working device. We are very satisfied with the outcome and will definitely keep the lessons and hardships we learned with us throughout our professional careers.

The first major accomplishment was having our individual modules independently working. We tested them individually and noted the results. As we slowly began piecing together the modules, we ran into some problems, but dealt with all of them properly. As we combined our modules, we did systematic debugging in order to find the sources of error or problem.

The most significant engineering accomplishments would deal with the constant problems we had with the 16-bit ADC correctly communicating with the microcontroller. Essentially, there was no data being sent to the microcontroller. However, we devised multiple solutions and contingency plans in order to have a successful device. After all the proper testing and execution, we ended up with a very successful deliverable.

### 6.2 Uncertainties

We had many difficulties along the way. First, we had about 75 different components (both through-hole and SMD) which was quite difficult to route. We created three versions of the PCB due to change in design

as well as cut traces. Second, as we tested the microphone probe from CERL by itself, it was had very low signal strength. It was too close to noise level and too small to resolve correctly.

One of the major difficulties was with the 16-bit ADC communication with the microcontroller. The ADC was not giving any data out for the microcontroller. Thus, we had to create another contingency plan to use the on-board 12-bit ADC. In addition to this, the microcontroller was having some memory issues.

Another major difficulty was that the LCD was drawing too much power. Because of this, we had to change the power supply to AA batteries instead of the original 4.2V LiPo batteries.

## 6.3 Ethics and Safety

### 6.3.1 Ethical Issues

As with all engineering projects we have to make sure that we are properly following the proper ethical guidelines. In order to check that we do not have any Ethical Issues Associated with out project we will review the IEEE code of ethics for any potential breach in ethics. Below are some of the issues associated with our specific project:

1. *to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment*

   Initially we were worried about what the implications of our project would be if used by an army research facility; but, after some discussions with the potential end users, we understood the basis of Professor Swenson and Dr. White's research. In general, the army generates a lot of noise and they are researching solutions to minimize the army's footprint in their area of operation. This research actually benefits the public because noise pollution and hearing damage is a daily challenge for some people.

2. *to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.* All group members have grown up in a diverse environment and will treat each other as well as the rest of our colleagues, fairly.

3. *to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.* While going over the ethical rules and guidelines, we have read and proofed each others work and therefore understand the IEEE code of Ethics. We will keep each other, as well as the rest of our peers, in check as we continue onward with our project.

### 6.3.2 Safety Issues

The biggest safety issue we had for our project is our use of a LiPo battery. LiPo batteries are one of the more unstable batteries in existence and has been known to heat up rapidly and catch fire if not used properly. In order to avoid this problem we have bought a pre-made battery management system board which has been tested and used by various individuals and is known to work properly. By having this intermediate module within our power module we will guarantee that our battery will not short of ignite due to human errors.

## 6.4 Future Developments

The three of us will be working on this project further during the summer. We would like to give Dr. Swenson and Dr. White a great deliverable and will be working with them in the future. Also, we also want to possibly pass on our project for next semesters senior design class so that they can improve this device. Below is a list of more future developments:

First, we would like to create a sleeker device with a hard metal or plastic casing. Due to time constraints, we were not able to create a proper casing. We would also like to include an instruction manual on how to use the device and how to create additional programs/functions in the software. Unfortunately, we were not able to integrate our device with Team 48s microphone probe. Hopefully in the near future we can combine our two projects to give Dr. Swenson and Dr. White an update version of what they already possess.

It was also mentioned that we should include a MicroSD module to save the data that was collected. Additionally, we want to successfully use a 16-bit ADC that was asked of us by the clients. Finally, we want better power distribution and have a rechargeable LiPo battery that outputs more current. We want to integrate an on/off switch instead of manually taking out batteries to power the device.

Overall, the three of us are looking forward to continue our work in the summer to give Dr. Swenson and Dr. White a great device.

# APPENDICES

## A Block Diagrams

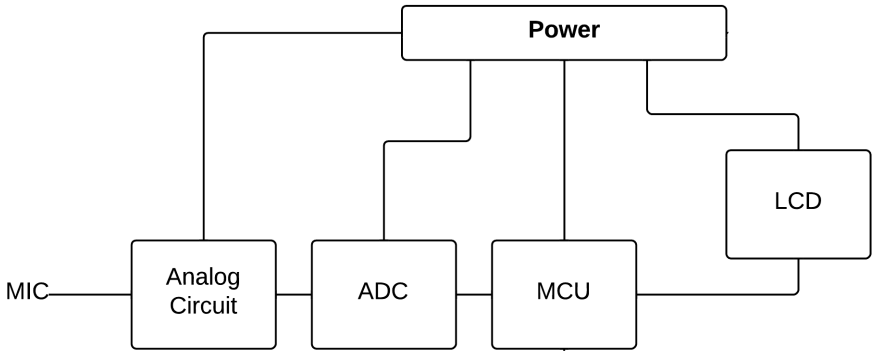


Figure 27: Top Level Block Diagram

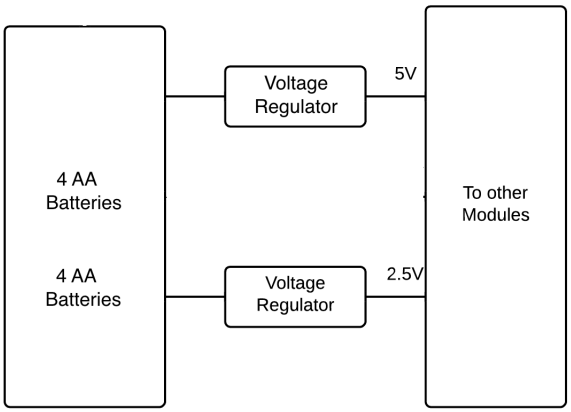


Figure 28: Power Supply Block Diagram

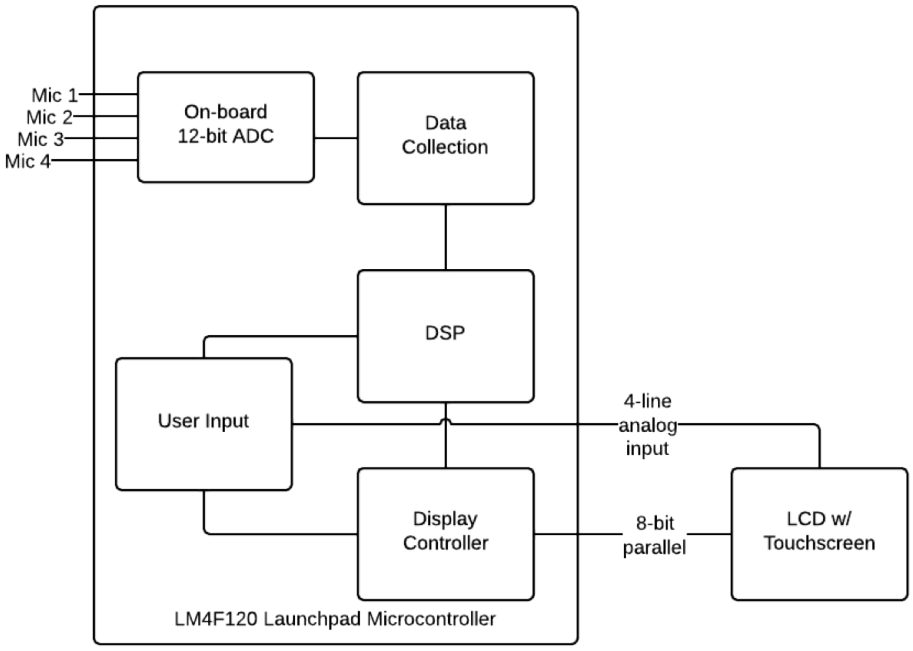


Figure 29: MCU Block Diagram

# B Requirements and Verification Tables

## B.1 Power

| Requirements | Verifications | Pass/Fail |
|---|---|---|
| 1) Check voltages for 5V, 3.3V, and 2.5V (Tolerance should be desired voltage ± 0.5V). | 1) Use a multimeter to check the outputs of the boost converter (5V) and the two voltage regulators (3.3V, 2.5V). | PASS |
| 2) Supply sufficient voltage each of the modules over time. (With a voltage ripple of ± 1.0 V ). | 2) Use an oscilloscope to see if there is steady voltage being given to individual module within the voltage ripple. | PASS |
| 3) Check individuals modules if receiving correct voltage. | 3) Check individual modules<br>a. Connect all modules to power supply<br>b. Verify if modules have the correct input voltage with a multimeter | PASS |
| 4) The Charging Circuit<br>i. The fully charged battery should be at maximum 4.2V<br>ii. The battery should only at minimum 3.0V at full discharge.<br><br>iii. With a charge rate of 1000mA, the battery (with rating of 2600mAH) should be fully charged in 2.6 hours (± 20 min). | 4) Charging Circuit<br>i. Using a multimeter, check the voltage of battery in initial testing (max 4.2V).<br>ii. Using a multimeter, check the voltage of battery after being connected to various loads (See Requirement 1) and must be above 3.0V<br>iii. Connect the battery to the wall outlet and check the amount of time it takes for the battery to be charged to a max 4.2V. | PASS |

**Changes Made**

The original design used a 3.7/4.2V LiPo battery. This battery, however, could not supply enough current to the LCD screen and would not turn on. The other modules powered fine; however, we chose to change the battery source to 8 easily replaceable AA batteries instead of the 3.7/4.2 V LiPo Battery. Thus, we removed the charging mechanism (even though this test passed with the LiPo battery). Also, we did not need a Vo=3.3V circuit anymore because we are not using the original clock chip in our first design.

## B.2 Filter

| **Requirements** | **Verifications** | Pass/Fail |
|---|---|---|
| 1) Check to see that the frequency response is similar to the simulation. | 1) Check several frequencies and match with simulations<br>a. Using a function generator and an oscilloscope check the filter output.<br>b. Measurements should start at 20Hz, 100Hz, 500Hz 1kHz then repeat every octave.<br>c. Conditions of sucess are shown below<br>- If datapoints deviates less than ±1dB from simulations<br>- If datapoints between 20Hz and 10kHz do not deviate more than 0.2 dB from each other | PASS |
| 2) Check to see that the filter gain is similar to the input with a small passband ripple. | 2) Input a 5kHz signal at 1Vpp and check the output. The output should also be roughly 1V ± 50mV. The passband ripple should be less than .1Vpp Afterwards input a 15kHz signal and check the output. The output should be roughly .6v ± 50mV. | PASS |

**Changes Made**

There were no changes made to this module.

## B.3 Amplifier

| **Requirements** | **Verifications** | Pass/Fail |
|---|---|---|
| 1) Verify that the signal is amplified after the circuit | 1) Determine if correct amplification<br>a. Connect input to known wave (square, sine) via the function generator.<br>b. Connect output to oscilloscope<br>c. Measure the amplitude of original wave first as a reference.<br>d. Measure the amplitude of wave after the amplifier<br>e. Make sure that the amplitude of original is multiplied by gain of circuit | PASS |
| 2) No additional noise is added to signal | 2) Determine if there is noise<br>a. Connect known wave and oscilloscope outline above (Steps 1a - 1b).<br>b. Visually look at noise of original wave<br>c. Visually compare to amount of noise after the amplifier.<br>d. Verify that there is no additional noise after the amplifier. | PASS |

**Changes Made**

Please note that this module of the design was added at a later date since the change in voltage from the microphone inputs were too small to detect on our LCD (window settings, etc.) Therefore, we decided to add this to see the respective change in voltages from the microphone probes.

## B.4 Microcontroller Unit

| **Requirements** | **Verifications** | Pass/Fail |
|---|---|---|
| Data Collection:<br>1) Proper ADC I2C to dual SPI Communication | Data Collection:<br>1) Ensure ADC I2C to dual SPI Communication<br>a. Connect Vref1 to AGND.<br>b. Connect Vref2 to AGND.<br>c. Connect VinL to AGND.<br>d. Connect VinR to Vcc.<br>e. Run Data Collection program once saving the left and right channel data.<br>f. Through the onboard ICDI and code composer studio view the data.<br>g. Verify left channel data to be 0x0000 and right channel data to be 0xFFFF. | FAIL |
| Digital Signal Processing:<br>2) FFT processes data correctly. | Digital Signal Processing:<br>2) For a predetermined set of audio data (constant tone of known frequency), verify FFT processing:<br>a. Insert 2048 samples data set of a 5kHz sine wave into memory<br>b. Run FFT code.<br>c. Extract output data set to computer using ICDI and code composer studios manually.<br>d. Plot data in Matlab and verify a frequency spike at 5kHz ± 100Hz. | PASS |
| Display Controller:<br>3) Data can be shown correctly on screen. | Display Controller:<br>3) Verify Data stored is shown on screen correctly (LCD screen must be verified first):<br>a. Manually change processed data of FFT to a progressive linear increase of values from 20Hz to 10kHz.<br>b. Run display controller code with LCD boosterpack attached.<br>c. Verify screen shows an increasing histogram type bar graph starting at 20Hz to 10kHz. | PASS |

**Changes Made**

Requirement number 1 failed since the ADC did not correctly communicate with the microcontroller. Thus, we had to revert to using the 12-bit on board ADC instead of the 16-bit ADC.

## B.5 ADC Module

| Requirements | Verifications | Pass/Fail |
|---|---|---|
| PCM1801:<br>1) ADC chip is powered correctly with appropriate parameters set. | PCM1801:<br>1) Ensure device is powered and output parameters are set:<br>a. Connect multimeter probe GND to ADC AGND.<br>b.Attach multimeter positive probe to ADC VCC. Voltage should be +5V nominal. Voltage should not be below<br>4.5V. Voltage should not exceed 5.5V.<br>c. Connect oscilloscope probe GND to ADC DGND<br>d.Attach multimeter positive probe to ADC VDD. Voltage should be +5V nominal. Voltage should not be below 4.5V. Voltage should not exceed 5.5V. | PASS |
| 2) ADC output signal is I2S format | 2) Ensure proper I2S output format:<br>a. Connect VREF1 to AGND.<br>b. Connect VREF2 to AGND.<br>c. Connect VINL to AGND.<br>d. Connect VINR to VCC.<br>e. Connect logic analyzer GND to ADC to ADC DGND.<br>f. Connect logic analyzer probe to ADC DOUT.<br>g. Connect function generator GND to ADC DGND<br>i. Set function generator to generate a 3MHz square wave with amplitude of 3.3V.<br>j. Connect ADC LRCK to ADC DGND. The logic analyzer should show 16 bits of audio data representing left channel which should be 0x0000<br>k. Connect ADC LRCK to 3.3V. The logic analyzer should show 16 bits of audio data representing right channel which should be 0xFFFF. | PASS |

**Changes Made**

We had to use the on board 12-bit ADC on the micro-controller instead of the original planned 16-bit ADC. The 16-bit ADC works on its own, hence the PASS for these requirements. However, when combined with the microcontroller and the other modules of the device, it did not work as planned.

## B.6 LCD

| Requirements | Verifications | Pass/Fail |
|---|---|---|
| 1) LCD powers on and displays properly | 1) Verify LCD screen operation with LaunchPad evaluation kit:<br>a. Plug in LCD screen boosterpack to Launchpad evaluation kit<br>b. Upload and run graphical library example code<br>c. Verify operation of graphical library example program as shown in figure 17. | PASS |
| 2) Touchscreen is responsive to finger or stylus touch | 2) Verify operation of touchscreen:<br>a. Plug LCD screen boosterpack to Launchpad evaluation kit<br>b. Run code<br>c. Touch screen with finger or stylus<br>d. Verify that the menus/ buttons have the correct function and display correctly on the screen. | PASS |
| 3) Touchscreen buttons change parameters (current screen shown, DSP parameters) depending on current parameters | 3) Verify push on screen change the proper parameters based on current parameters:<br>a. Manually change the default parameters<br>b. Run User Input program<br>c. Depress each menu button<br>d. Check parameters afterwards using ICDI and code composer to check if the correct parameters changed to the correct values.<br>e. Repeat as necessary. | PASS |

**Changes Made**

We used to have just an LCD with physical buttons as the user interface. However, we changed this in order to incorporate the touch screen functionality of the LCD boosterpack. The menus and options are now displayed on the LCD screen and can be depressed by touching the screen with a finger or stylus.

# C Schematics and PCB Layouts

## C.1 Power Supply

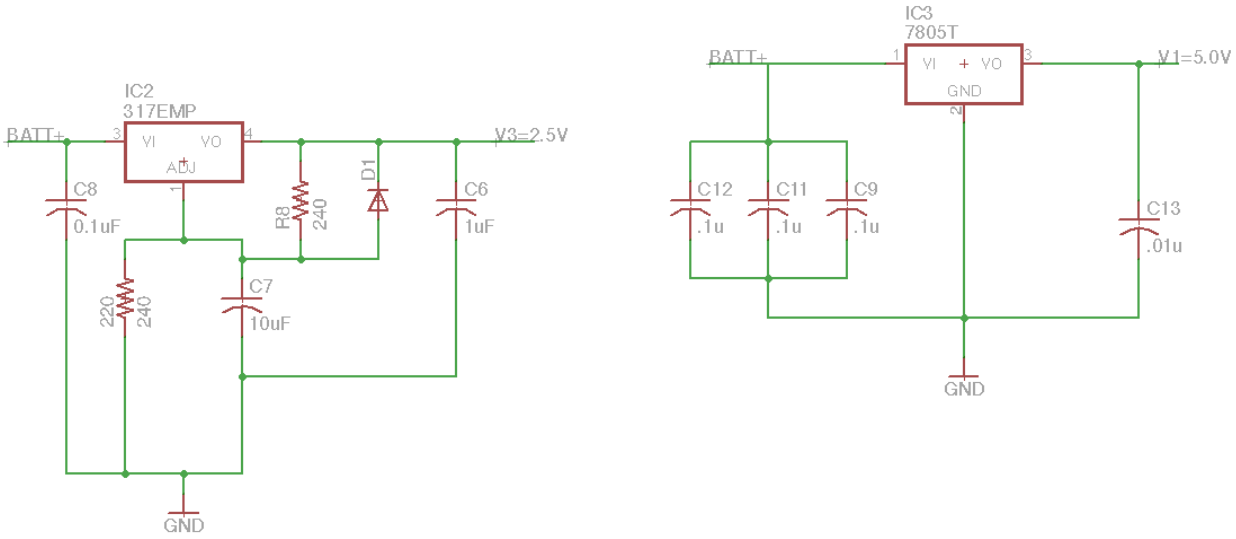


Figure 30: Power Supply Schematics

## C.2 Analog

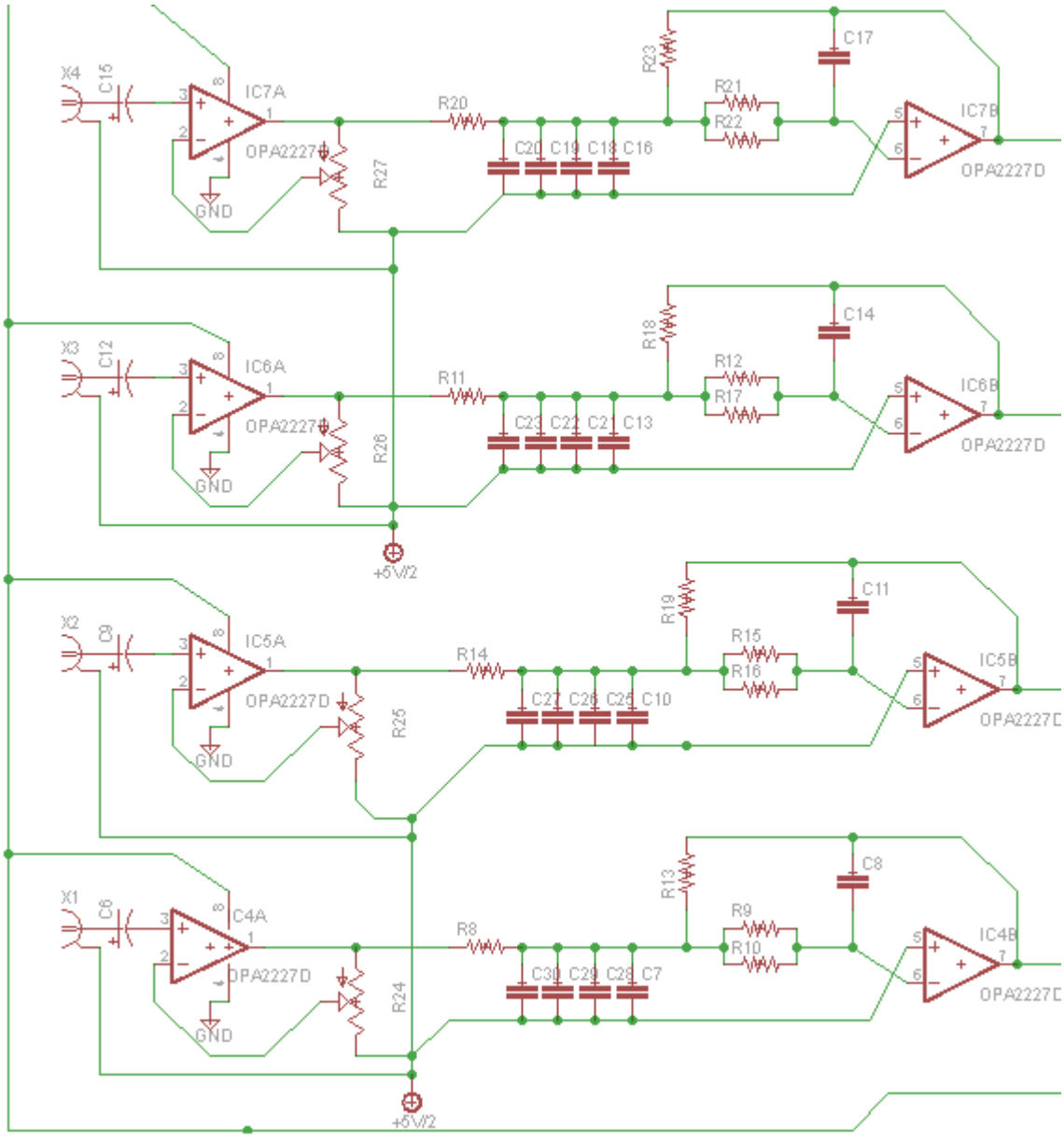


Figure 31: Analog Schematic 4 Input

## C.3 Microcontroller

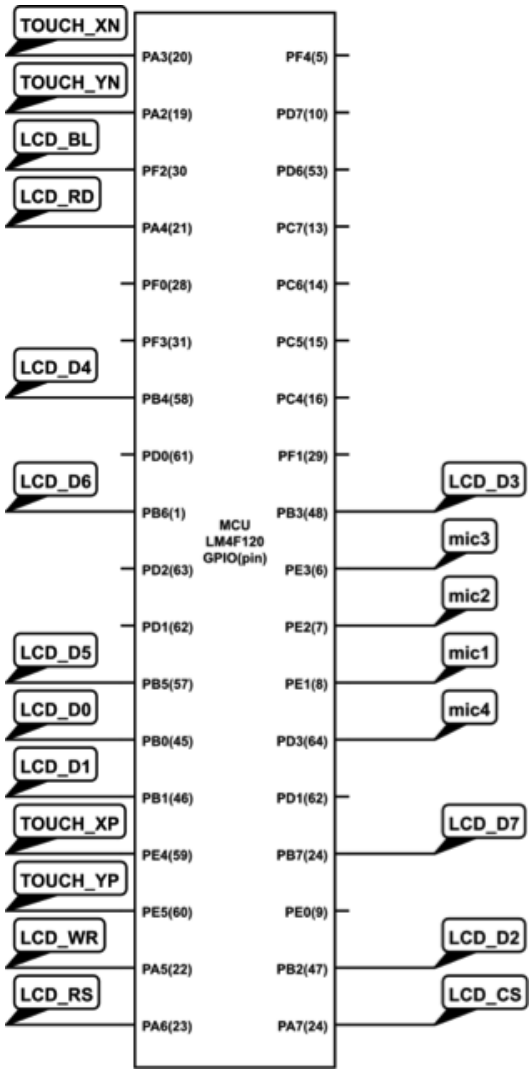


Figure 32: Final Pin Configurations

## C.4 Final PCB

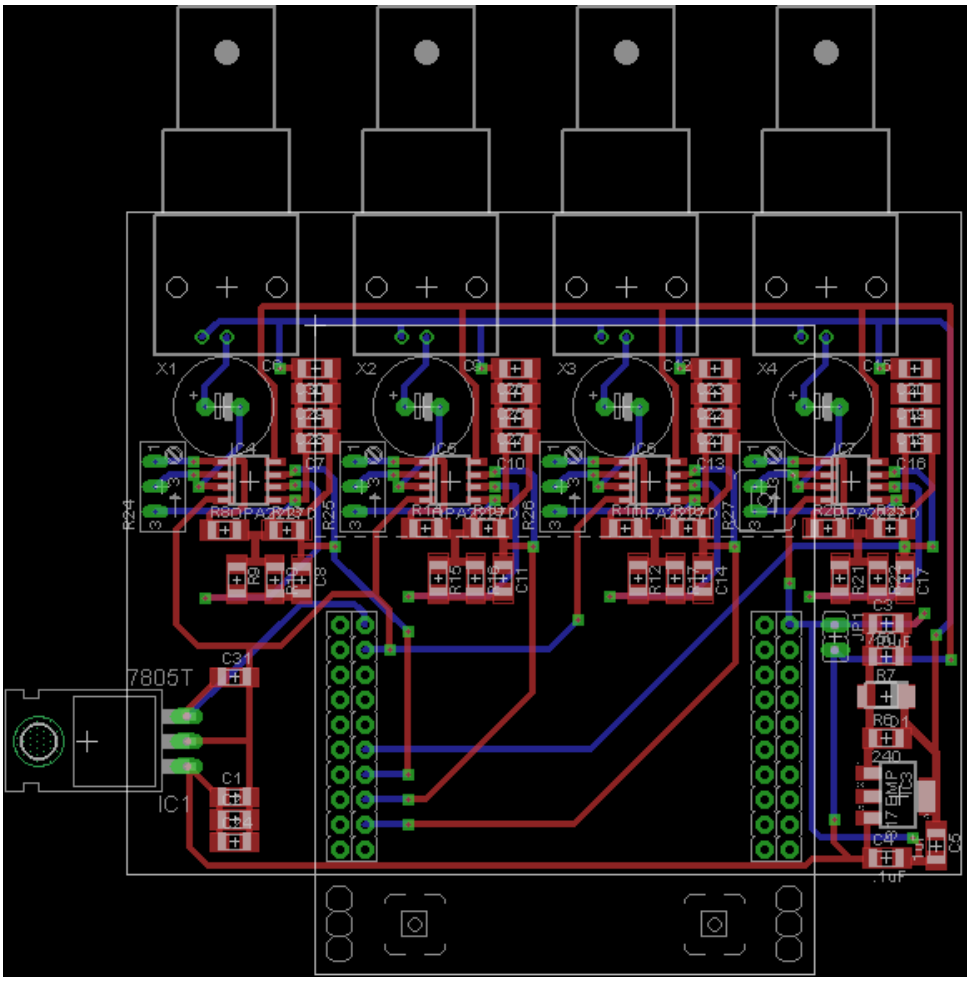


Figure 33: Final PCB Layout

# D References


1. "LM317." Texas Instruments. Texas Instruments, n.d. Web. 22 Feb 2013.
   *http://www.ti.com/lit/ds/symlink/lm317.pdf*
2. "LM7805." Texas Instruments. Texas Instruments, n.d. Web. 01 Apr 2013.
   *http://www.ti.com/lit/ds/symlink/lm7805.pdf*
3. "USB/DC Lithium Polymer battery charger." Adafruit. Adafruit. Web. 23 Feb 2013.
   *https://www.adafruit.com/products/280*
4. Stellaris LM4F120 LaunchPad Evaluation Board User Manual, Austin, TX, 2010. [Online].
   *http://www.ti.com/litv/pdf/spmu289a*
5. Stellaris LaunchPad LCD Boosterpack EB-LM4F120-L35 Users Guide, Futian District, Shenzhen, PRC. [Online].
   *http://www.kentecdisplay.com/uploads/soft/Products_spec/EB-LM4F120-L35_UserGuide_04.pdf*
6. Single-Ended Analog-Input 16-Bit Stereo Analog-to-Digital Converter, Dallas, TX, 2007. [Online].
   *http://www.mouser.com/ds/2/405/sbas131c-114090.pdf*
7. Stellaris LM4F120H5QR Microcontroller Data Sheet, Austin, TX, 2013. [Online].
   *http://www.ti.com/lit/gpn/lm4f120h5qr*
8. Stellaris Peripheral Driver Library Users Guide, Austin, TX, 2012. [Online]
   *http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=spmu019o&fileType=pdf*
9. Stellaris Graphics Library Users Guide, Austin, TX, 2012. [Online]
   *http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=spmu018o&fileType=pdf*
10. Dual-SPI Emulating I2S on Stellaris LM4F MCUs, Austin, TX, 2012. [Online]
    *http://www.ti.com/general/docs/lit/getliterature.tsp?literatureNumber=spma042a&fileType=pdf*
11. Using the CMSIS DSP Library in Code Composer Studio for Stellaris, Austin, TX, 2012. [Online]
    *http://www.ti.com/lit/an/spma041b/spma041b.pdf*