University of Nebraska - Lincoln

Computer Engineering Senior Design Project

Wireless In-Ear Audio Monitor

Team Stonehenge:
Erin Bartholomew
Paul Bauer
Nate Lowry
Sabina Manandhar

May 4, 2010

Contents

List of Figures

1 Motivation

In live band performances today, audio levels for playback monitors are controlled through a central audio mixing device. This device often controls multiple audio levels and is difficult to locate so that each band member has easy access. A band member can only change the audio levels for his in-ear monitor between songs. Such a setup becomes problematic when a band member needs to adjust the volume on a particular instrument because that instrument is being featured for a specific part of the song they are playing.

In fact, two of our team members have direct experience with this problem. Nate Lowry and Paul Bauer often perform in various local venues with other members of their group. They have difficulty being able to adjust playback monitors during songs due to the location of the single mixer that controls those monitors. After doing some research, the group discovered a solution that included a 48 channel mixer and several other components that were beyond the needs of the group. This solution was also over $100,000, which is quite expensive.

Audio signals in playback monitors also need to be practically synchronized with the real music to be helpful. Even 50 milliseconds can throw a musician off when he is playing. Keeping delay very small made more difficult due to differences in audio interfaces, from digital to analog. To convert between multiple interfaces would take more time than is allowable for audio delay.

Thus, the motivation for our Senior Design project was to create a way to more discretely and conveniently control the audio of instruments on playback monitors during live band performances. Our team was also able to achieve the goal of making the solution affordable so it is a feasible option for musical groups to implement.

This report will discuss the following things:

- Our proposed system
- Specific implementation details
- Experiences our team has that made this project a success
- Challenges that we faced during the project
- Cost breakdown for the final product

2 Our System

For this project, we created a Wireless Monitor System which takes input signals from audio sources and alters the volumes of these signals based on input from mobile controllers such as the IPod, Zune, and Motorola Android. Audio signals from audio sources such as microphones and guitars are given as input. These signals are processed by an audio controller board which adjusts the volume based on user input. The resulting signals are summed and output to a wireless audio transmitter. The audio-controller board is controlled by an Arduino micro-controller, which also hosts a private Web Site accessible by mobile devices to adjust volume levels. The adjusted volume is then experienced by the end user with the help of headphones or a receiver.

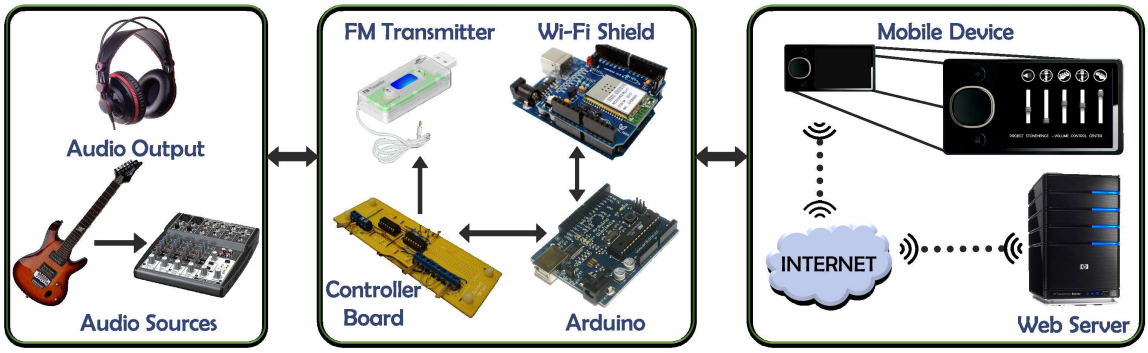

Figure 1: Schematic

The main components used in the system are as follows:

1. Audio Mixer: It is an electronic device which combines or mixes two or more audio signals to produce combined output signal. We will use audio mixer to combine incoming sounds from the audio sources and feed the combined output signal to audio controller board.

2. Audio Controller Board: It is a circuit board that will process the incoming audio signals based on its characteristics and output it to wireless audio transmitter. It will be connected to Controlling computer via a serial port.

3. Web Server: The web server houses the JavaScript for our Web Site. The server sends amplification information to an Arduino device through a wireless connection. The Arduino then feeds that information to the audio controller board, where the sound is amplified and sent to the user.

4. FM Transmitter: The FM transmitter will broadcast the amplified audio output so the user is able to receive the output for the in-ear audio monitor.

5. Mobile controllers: (Zune/IPod/Droid): These are portable media-player that will be used as volume sliders in our system. The device is facilitated with Wi-Fi which we will use to communicate wirelessly with the web server to get desired volume levels.

6. Headphones with FM Receiver: This represents the receiving end of communication channel which receive decoded messages send from the sender. The final output is the adjusted volume processed by our system can be listened by users using headphones or receiver.

3 Implementation

In this section, we follow the logical flow of the system from the audio source to the user's in-ear monitor. The components and interfaces between them are explained.

Usable audio sources include any musical instrument or microphone that is able to output to 1/4" TRS (Tip, Ring, Sleeve) connector. The 1/4" cable will be the input to the audio controller board. Our primary focus was the electric guitar, though an XLR microphone can be used with the proper cable adapter.

The audio controller board is a circuit board designed and assembled by the team. It has 4-1/4" inputs and 1-1/4" output. The main function of the board is digital volume control, provided by two PGA2311 chips. The PGA2311 chips change the amplitude, or volume, of the analog input signals based on digital input signals from an Arduino micro-controller. These resulting signals are summed using a simple single operational amplifier summing circuit, then output to the 1/4" output port. The audio controller board is examined with more detail in a future section.

For this specific use (wireless in-ear monitor system) a wireless transmitter is used for output, though any set of speakers or headphones with a 1/4" plug may be used. The transmitter may be a UHF wireless professional audio transmitter or a cheaper FM transmitter. The user has the appropriate wireless receiver on their person. This could be a UHF receiver, a standard FM receiver or even a pocket MP3 player tuned to the proper frequency. For

our project, we used a Sansa Clip MP3 player which contains a FM receiver.

Another component of the system is the mobile controller. This can be any internet enabled cell phone or other wireless device. The mobile controller accesses a web site hosted on a private server. This web site has volume controls that allow the increase, decrease or muting of the different audio channels. The interface to the web site is shown in Figure 2, below.

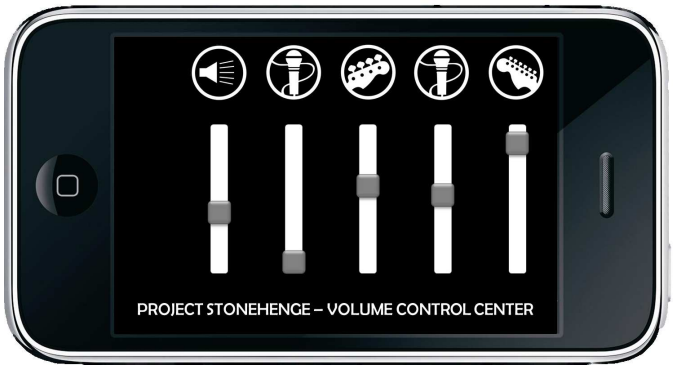

Figure 2: Web Interface

The interface was constructed with HTML and JavaScript (jQuery), making JavaScript is a requirement for any browser used. Proprietary frameworks like Flash were not looked at because of further limitations it would impose. The interface includes sliders for each channel of audio, these sliders control the volume level of each channel. The interface also shows an icon representing what the channel is controlling - guitar or microphone. The final part of the interface is a master volume slider that affects all channels.

A mobile controller interfaces with the controlling web site, hosted on a private server. The server converts the volume slider positions to HTTP requests that are then sent to the Arduino micro-controller, which then sends the resulting digital commands to the audio controller board. This means much of the web server load is handled by the private server, rather than the Arduino. This is important, as the Arduino is not powerful enough to handle too much load.

4 Experience

Our experience with these types of systems starts with two of our members playing in a local band. They deal with monitor levels on a monthly basis,

but still struggle to find a system that works. They were the default voice for features and usefulness of the system.

All of the group members had some experience with the components. We had all worked with integrated circuits, web servers, mobile devices, and basic web site creation. Our Electrical Engineering labs used many of the smaller components such as amplifiers, potentiometers, and digital logic chips. We were able to use this experience and knowledge of the field to select the proper parts and assemble them correctly.

This project provided a natural division of work. The main components (the mobile controller, audio controller board, and web server) are very loosely coupled. They interact with each other on a very limited basis in a very defined way. The interfaces for the audio signals are always through a 1/4" audio jack. The private server interacts with the controller board through a standard interface designed by the team. Finally, the web interfaces use standard HTTP protocols as well as a standard web browser interface.

This de-coupling allowed the team to divide the work into areas of user interface design, networking and circuit construction. We were able to leverage the individual strengths of each member in these areas.

5 Final Progress

Our final product included all of the components for two of the channels. We used guitar, microphone, and a computer headphone jack for inputs. We also used a FM transmitter for the output.

5.1 Audio Controller Board

At the heart of the audio controller board are two PGA 2311 stereo volume control chips. These two chips provide the ability to digitally control the volume of up to 4 input channels. Digital commands sent by the Arduino microcontroller provide the volume control in our solution. There are three command lines sent by the Arduino: Chip Select (CS), Serial Clock (SCLK), and Serial Input (SDI).

PGA 2311 volume levels are set by supplying an eight bit command per channel (sent on the SDI line). These eight bits are each locked in on the rising serial clock command (SCLK). This rise on SCLK is ignored unless

Chip Select (CS, actually NOT CS on PGA 2311 chips) is active. The figure below shows a typical volume-setting cycle for the PGA 2311 chip. Note that the right channel is read first (most significant bit R7, specifically), followed then by the left channel. Also note that we do NOT use stereo amplification in our solution. Therefore we are able to use the right channel as our first mono-amplified channel and the left channel as our second mono-amplified channel, providing us with two channels per PGA 2311 chip.

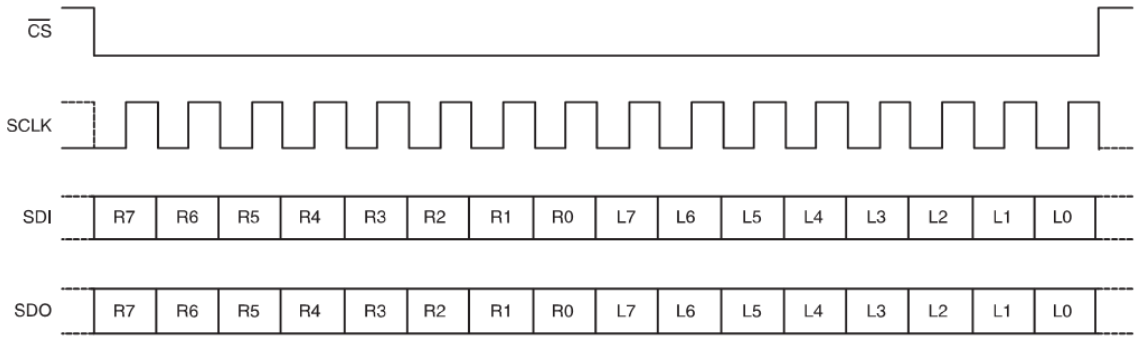

Figure 3: Digital commands to set volume of a PGA 2311 chip

The above figure also shows an SDO command, representing serial out of the PGA 2311 chip. Multiple chips can be daisy-chained by tying a second chip's SDI line to the SDO of the first chip (the two chips sharing CS and SCLK). The second chip's volume commands are read first in this arrangement. By daisy chaining two chips together a user is able to provide volume control for four channels (two channels per chip).

Our implementation of a PGA 2311 volume control system is shown figure 4 on the next page.

The signal outputs from the PGA 2311 chips are sent into a summing circuit, which allows us to sum the 4 signals into one resulting output signal. Figure 5, also on the next page, shows our simple summing circuit configuration.

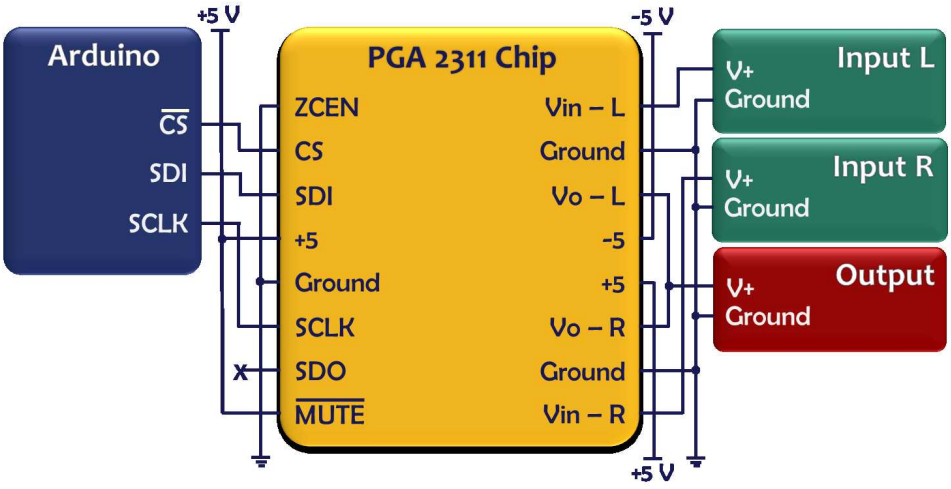

Figure 4: Digital commands to set volume of a PGA 2311 chip

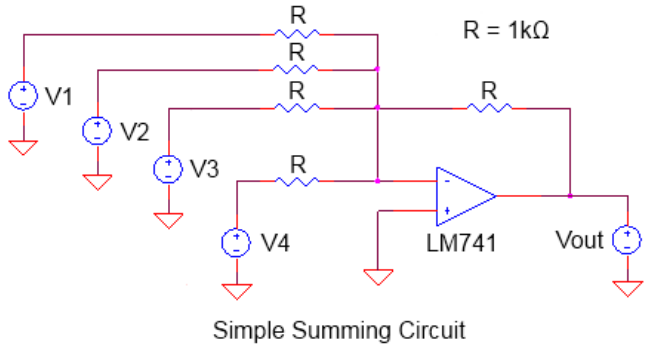

Figure 5: Summing Circuit

5.2 Web-Based Application

Since the Arduino has limited capacity, we are using WiShield (hardware that connect to wireless network) and WiServer to process the data and send only tiny amount of data containing commands for the volume control chip to the Arduino board. Our first attempt was to setup a PHP server and run it on the Arduino, but after further research, we realized PHP will not work with the Arduino web server.

We decided to use a separate server with AJAX and JavaScript. We set up the Arduino web server and have AJAX working. The interface of Web Site is written in HTML and JavaScript allowing it render in any web browser. The team is leveraging the jQuery JavaScript framework to handle much of

the site's UI. The HTML file contains a script which will link it to external java script. The external java script will have all the necessary user-interface and send backs only the required volume in plain text to the HTML file as depicted in figure below.

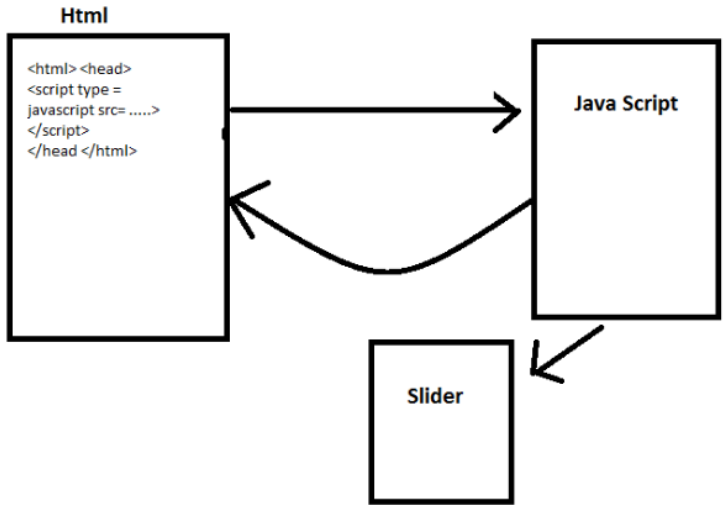

Figure 6: Prototype for HTML and JavaScript

5.2.1 User Interface

The user interface is a static Web Site which points to the Arduino. It consists of five sliders:

- Master volume
- Microphone
- Guitar
- Bass
- Microphone

For user convenience, we have added the ability to hide channels. The user can display only the channel he is using and hide the rest. We used jQuery to make the user interface because it is easier to use and gives more flexibility to developers. It uses some familiar JavaScript methods, but also has built-in functions and methods that are easier to use. The user-interface can be controlled from any mobile device which has Internet access and supports

JavaScript. Once the mobile device is connected to the controlling Web Site, the user can use the sliders to control the volume of desired channels. We have also implemented password protection to the Web Site so that only authorized users will be able to control the channels.

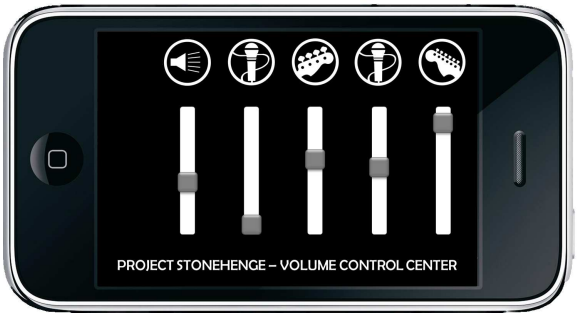

Figure 7: User Interface

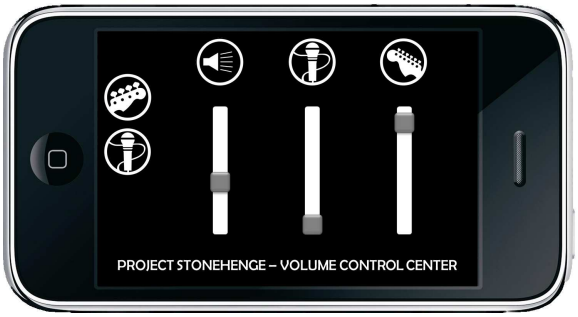

Figure 8: User Interface with Collapsed Sliders

5.2.2 WiShield

This is a piece of hardware that enables wireless connectivity to the Arduino. The board includes stack-through headers to allow access to unused pins and a 9-pin breakout header space for prototyping. To set-up wireless configuration, we downloaded the WiShield software and WiShield library. We had to configure wireless parameters like SSID and security for wireless network. Since WiShiled does not support DHCP, we have to select a static IP address. The schematic of the WiShield is shown on the next page.

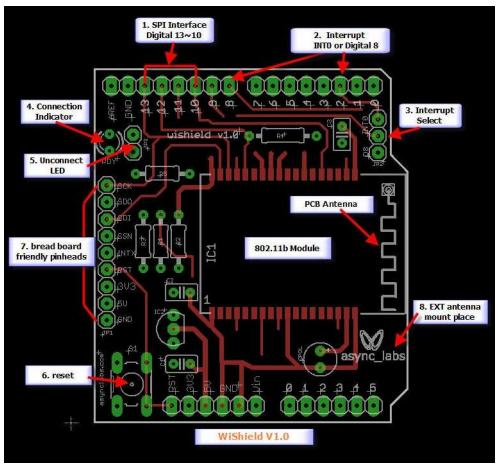

Figure 9: WiShield Schematic

6 Challenges

6.1 Delay

For controlling any audio signals one sensitive issue is delay. We needed to account for the audio timing issues between different components and delay due to the private web-server.

6.1.1 Sound Delay

Sound delay is the most critical form of delay in our system. Sound delay between signal input and output can be disastrous for a performer. A delay of only a few hundred milliseconds can have a huge impact when trying to keep members of a band in time with each other. We made efforts to minimize delay by keeping all audio signals in their analog forms throughout the amplification process.

Once the system was completed, we measured delay using Audacity, an open source software audio. The input signal is a generated sine wave that is fed through the controller board and back into the computer. There is additional delay that is introduced by the computer hardware, so we also recorded a base amount of delay by running the same sine wave directly from the computer's output to the input. The sound delay is the difference between the base case of delay and the delay from the output to our system. After several trials, we recorded an average delay of approximately 7.3 milliseconds which is definitely an acceptable amount of delay during a performance.

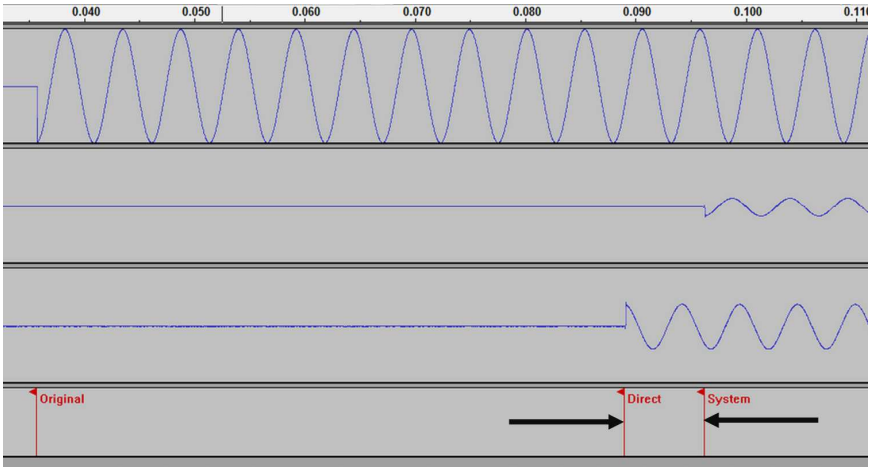

Figure 10: Input and Output Delay Measurements using Audacity

6.1.2 Web Delay

Web delay is less of a concern than sound delay, but is essential to minimize for a snappy system. To measure web delay, we used Wireshark, an open source software program used to capture structure of networking protocols and displays details of the packet data. The main purpose of using Wireshark is to troubleshoot network problems and examine security problems. Time delay through the web-server can be tested using this software. In testing the web delay, we measured the time difference between the RESTful packets sent by the web server to the Ardunio and the acknowledgment sent back by the Arduino. Using this information, we recorded an average web delay of approximately 0.5 seconds.

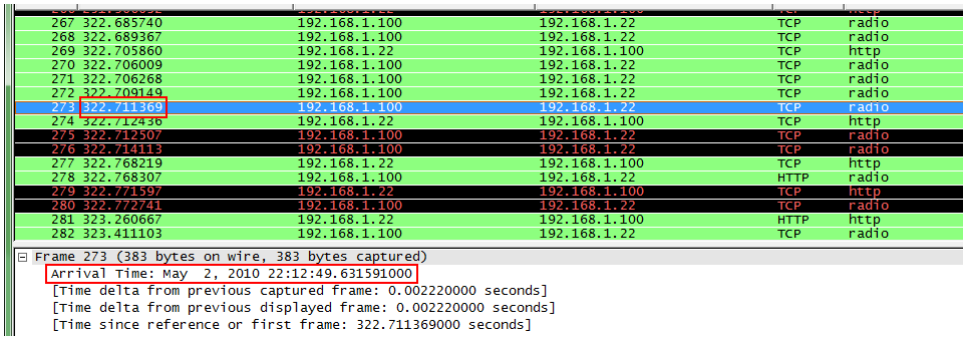

Figure 11: Screenshot of Wireshark Tool Recording Web Delay

6.2 Distortion

Another design concern that our team monitored is that of distortion. For our project, we were not concerned with minor distortion as the only person listening to the output of our system will be the performer. However, we needed to make sure the performer was clearly able to hear all parts through the monitor.

We had several problems with distortion when trying to sum the output signals, requiring several iterations of summing circuits before arriving at our final one. Our final solution is a simple current to voltage converter.

7 Costs

One of the major factors in our design was keeping the overall cost low. Here are our costs and comparable costs of similar systems. Because afford ability

Arduino Microcontroller	$50.00
Arduino WiShield	$30.00
Project Box	$6.00
Arduino ScrewShield	$12.00
2x PGA 2311 Chips	$32.00
Other Elec. Components	$20.00
Total	$134.00

Table 1: Cost Breakdown of Our System

is a large motivation for this project, we also performed a price comparison to similar products that are already in the market. The chart shown in figure 12, on the following page.

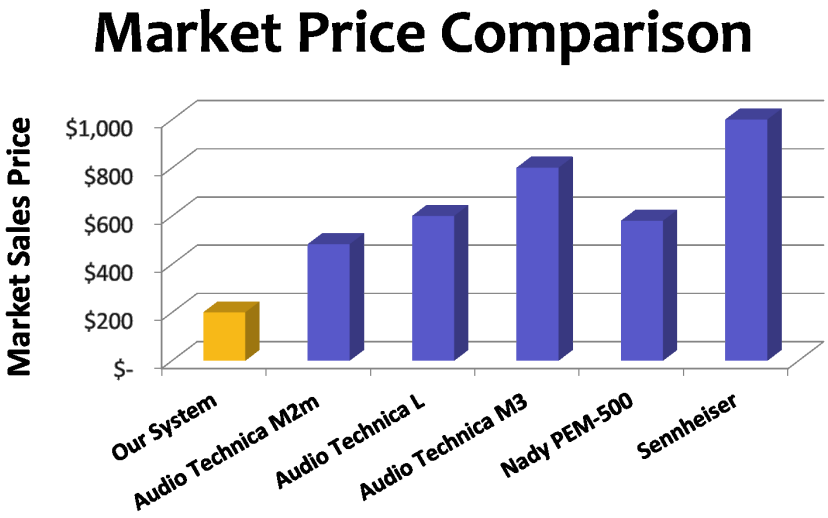

Figure 12: Market Price Comparison

8 Conclusion

We have implemented all major features of our proposed system. We have also configured a Project Box to house the audio controller board and Arduino device. The project box has openings for audio inputs and outputs. The project box will make our controller board configuration more secure and portable. This is an imperative feature of a wireless audio monitoring system because bands must be able to take the device to various performance locations easily.

8.1 Future Plans

We have learned a lot about the wireless capabilities of the Arduino device and the WiShield. The WiShield is first generation technology, making it unreliable and difficult to work with. Nate Lowry and Paul Bauer intend to continue working on this project, likely removing the wireless component until a more reliable solution can be found.