sound360

an immersive sound experience

David Baker & Marcus Pan

baked@seas.upenn.edu | mpanj@seas.upenn.edu

Advised by Dr. Rahul Mangharam and David Glanzman

University of Pennsylvania

ESE 350: Embedded Systems

Spring 2015

Table of Contents

Abstract

We designed sound360, a highly directional surround sound system for a living room that enriches the TV viewing experience. If we modulate an audio signal using pulse width modulation at an ultrasonic frequency, the nonlinearities in the air will demodulate it back to the audible domain. Since the audio input is now carried by an ultrasonic frequency, it is much more directed. Once we can generate highly directional sound with an mbed, we can create cool psychoacoustic effects. We designed a 'sound dome' that has planes of ultrasonic transducers pointing at 6 different angles. The 6 narrow beams of sound are reflected off panels at a centrally located viewer/listener. We also came up with a signal protocol using the standard stereo audio jack so that media content providers can now couple direction data with the audio data in real-time. This adds a new dimension to the media viewing experience.

Introduction

Current audio surround sound systems enable about 5 different channels of sound across different speakers. These speakers are placed around a living room and are pointed toward a user, as illustrated in figure 1.

This configuration requires a separate speaker for each channel, and the directivity of sound is still limited by the low directionality of the waves in the audible frequency range.

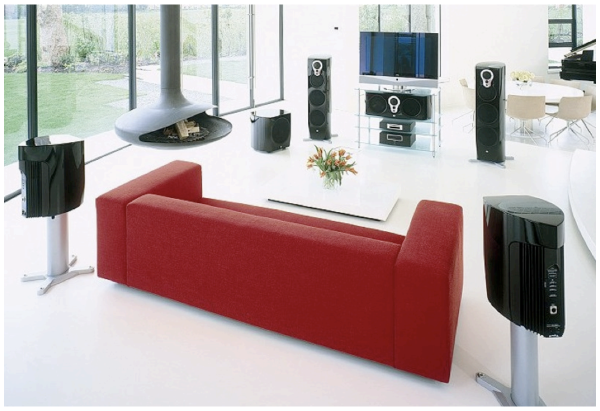


Figure 1. Typical setup of a surround sound system. ***Source: http://gizmodo.com/5286069/do-you-have-a-surround-sound-setup*****.**

We designed a new speaker setup that employs directional audio to produce sound signals that is transmitted in a narrow beam. The sound source consists of one centralized speaker array with 6 different sub-arrays that are angled at different directions. This works best with a reflector boards set up at the corresponding angles, directed toward the user. Finally, we also came up with a protocol to couple angular input, the direction the audio comes from, with audio input in a given media clip. This system improves current surround sound systems significantly by eliminating the need for multiple speakers and enhancing the directionality of sound.

Principles of Directional Audio

The directivity of sound is related to the ratio of the wavelength of sound to the speaker length[1]. We thus transmit the signals at a high ultrasonic frequency (40 kHz) to achieve high directionality. Every medium that the sound travels in can be characterized by a nonlinear parameter. In our case, the nonlinear parameter of air demodulates the sound back to the frequency domain if we transmit it as a pulse width modulated (PWM) signal. This modulation technique is easily done on a microcontroller.

Figure 2 illustrates the directivity of sound at audible and ultrasonic frequencies.

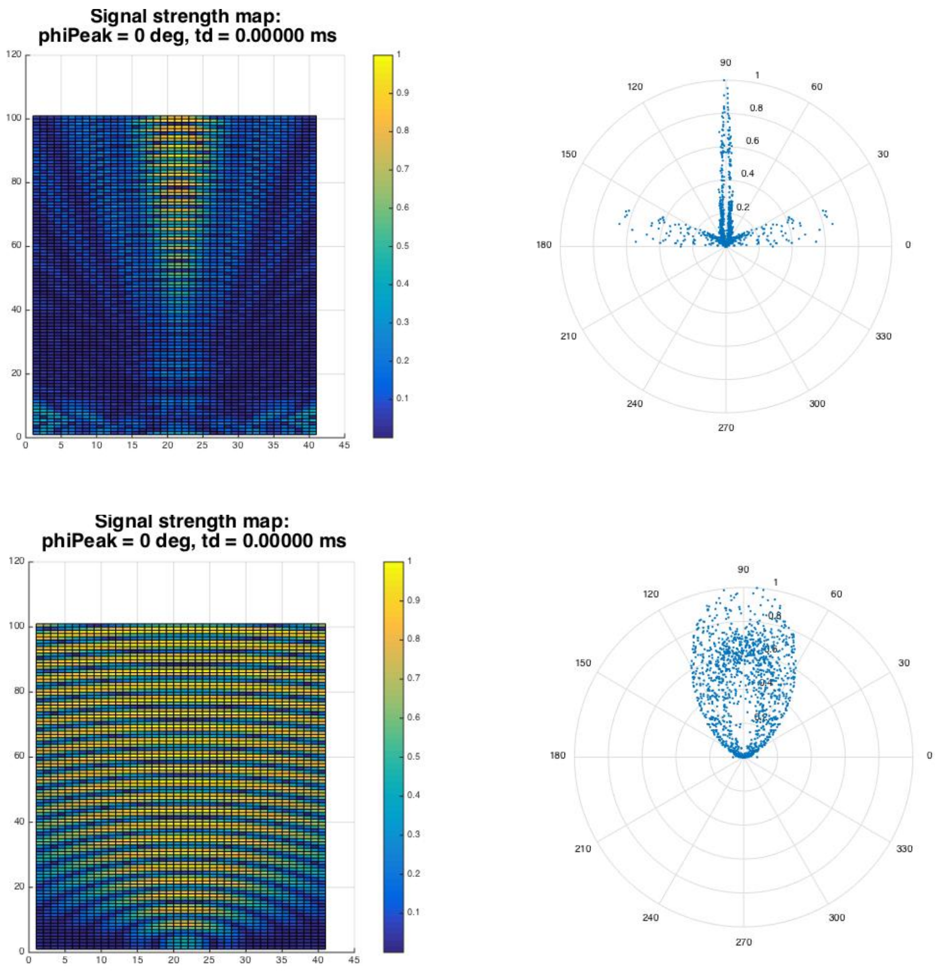


Figure 2. Matlab model displays the directivity of a 40 kHz audio signal (top) and a 4 kHz audio signal (bottom).

[1] F. J. Pompei, "Sound from Ultrasound: The Parametric Array as an Audible Sound Source," Ph.D. dissertation, Prog. of Media Arts & Sciences, MIT, Cambridge, MA, 2002.

System Architecture

Figure 3. System flow diagram.

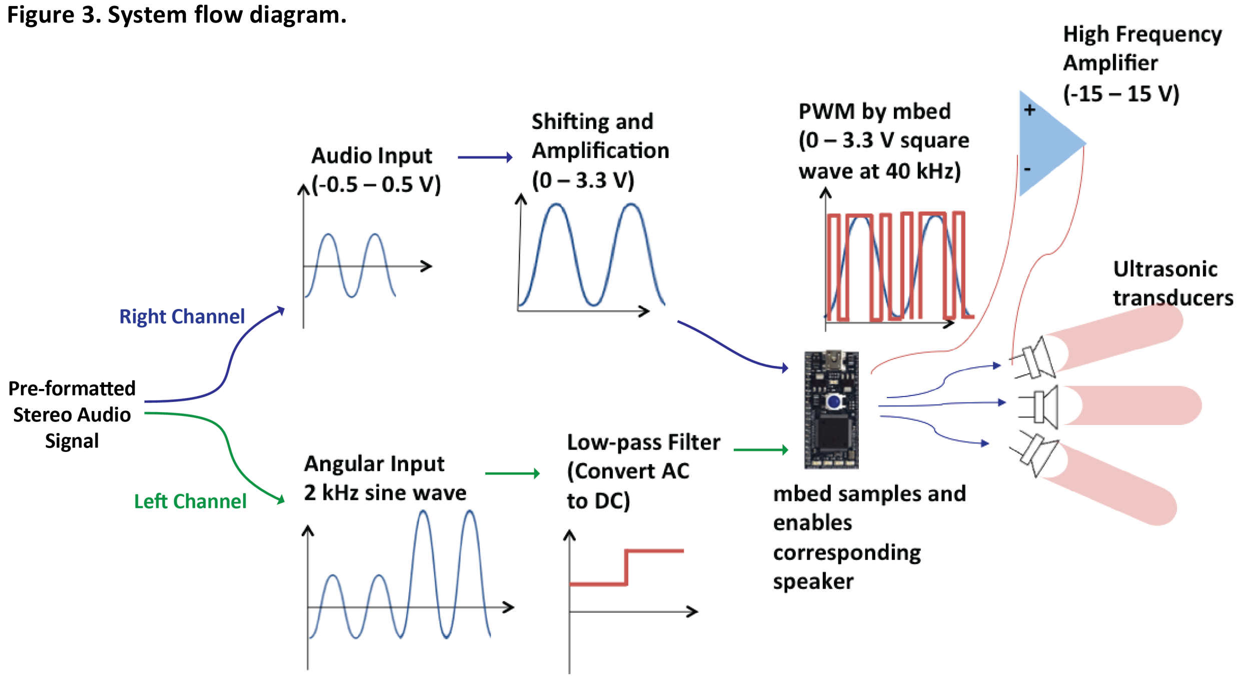

High Frequency Amplifier (-15 – 15 V)
Audio Input (-0.5 – 0.5 V)
Shifting and Amplification (0 – 3.3 V)
PWM by mbed (0 – 3.3 V square wave at 40 kHz)
Ultrasonic transducers
Right Channel
Pre-formatted Stereo Audio Signal
Left Channel
Angular Input 2 kHz sine wave
Low-pass Filter (Convert AC to DC)
mbed samples and enables corresponding speaker

Figure 3 shows the overall flow of our system from input signal to output sound. Now we'll break it down into sections. We begin with a standard 3.5 mm stereo auxiliary cable. The cable has one end hooked up to a source, say a computer, which already has our pre-formatted audio signal. We will cover how we generate our desired format in a later section. The pre-formatted audio signal is really two signals—a left channel and a right channel. We send the video soundtrack over the right channel, and the angular input signal over the left channel.

Audio Input – Right Channel

The soundtrack signal coming out of the computer is AC (centered at 0 V) and has an amplitude of about 0.5 V, but this amplitude varies by source and volume of the source. To perform an analog-to-digital conversion on the mbed (with maximal resolution), we need the signal to vary from 0 to 3.3 V (centered at 1.65 V with an amplitude of 1.65 V). It is very

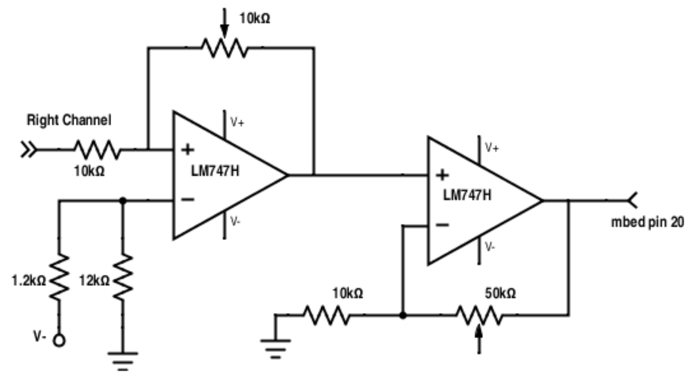


Figure 4. Audio input shift and amplification circuit.

important we span the entire 3.3 V because the duty cycle of the 40 kHz output signal is directly correlated to the ADC value read in, and this effects the final volume heard by the viewer. We accomplish this shift and amplification with the circuit shown in figure 4. Because different sources were so variable in their amplitude, we use potentiometers in our feedback loops to vary the gain accordingly. The first stage is an inverting summer that shifts the voltage up, and the second stage is a non-inverting amplifier that amplifies it.

Angular Input – Left Channel

The angle—or more precisely which of the six directional modules—the sound will project from is specified on the left channel of the auxiliary cable. A 2 kHz sine wave with varying amplitude (set in the pre-formatting) is sent over this channel, where it is first amplified such that its amplitude is about 5 V. This sine wave is then clipped with a Zener diode so that only positive voltages may pass. Then, a simple RC low-pass filter extracts a DC level that is directly correlated with the amplitude of the 2 kHz sine wave. This voltage is then sent to pin 15 of the mbed. This circuit (after the amplification stage) is shown in figure 5. The voltage that is read in by ADC specifies the desired angle of the accompanying audio input.

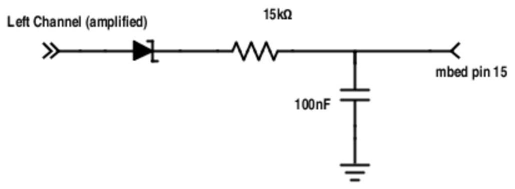


Figure 5. Filter that extracts a DC level from an AC sinusoid to be used for angular input.

mbed Processing

A single mbed LPC1768 functions as a Digital Signal Processor (DSP) in our system. It samples both the angular input and audio input at about 200 kHz. This is sufficiently fast sampling for audio signals that are limited to 20 kHz. The mbed then performs PWM on the audio input at an ultrasonic frequency. It sets up a 40 kHz square wave and varies the duty cycle with the amplitude of the audio input. The signal transmitted in this form then gets demodulated back to the audible domain as it passes through nonlinearities in the air.

We could not rely on the mbed PWM libraries for such a dynamic change in the duty cycle, so we programmed the settings registers directly. To do so, we had to referred to the following documents:

1. mbed SDK code source (https://github.com/mbedmicro/mbed)
2. mbed LPC1768 user manual (http://www.nxp.com/documents/user_manual/UM10360.pdf)
3. Helpful blog post (http://msys-mv.blogspot.com/2011/01/using-pwm-on-lpc17xx.html)

The following section walks through the low-level C code we wrote for the corresponding functions. Most of the register names can be found in the following file in the mbed SDK:

`libraries/mbed/targets/cmsis/TARGET_NXP/TARGET_LPC176X/LPC17xx.h`

a) `setupPWM()`

The first step is to power the PWM module, in the PCONP register:

```
LPC_SC->PCONP |= 1 << 6; //power PWM
```

Then we select the default clock frequency (96 MHz) for the PWM counter:

```
LPC_SC->PCLKSEL0 |= 1 << 12; //PWMclk = cpu clk (96MHz)
```

Next we configure the PWM registers that control the duty cycle and frequency. Its basic configuration is as follows:

TCNT increments at 96 MHz
When TCNT == MR0 : output HI;
When TCNT == MR1 : output LO;

To achieve a 40 kHz frequency, and initialize a 50% duty cycle we thus set:

```
LPC_PWM1->MR0 = 2400; //output HI every 2400 ticks(40 khz)
LPC_PWM1->MR1 = 1200; //output LO when TCNT = MR1, init to 50%
```

Finally we enable the PWM at corresponding pin26:

```
LPC_PWM1->TCR = 9;//enable pwm
LPC_PWM1->PCR = 1<<9;//enable output on pin26
```

b) `setupADC()`

This section documents the code to setup the ADC conversion.

First we power on the ADC module in the PCONP register, and select an appropriate clock speed (one conversion takes about 60 cycles):

```
LPC_SC->PCONP |= 1 << 12;//power ADC
LPC_SC->PCLKSEL0 |= 3 << 24;//ADCclk = cpuclk/8 = 12 MHz
```

Next we select ADC mode for our input pin20 (audio input) and pin15 (angular input):

```
LPC_PINCON->PINSEL3 |= 3 << 30;//select AD mode fr pin20 (audio input)
LPC_PINCON->PINSEL1 |= 1 << 14; //pin15 is AD mode (angular input);
```

Finally we configure the settings to enable the ADC:

```
LPC_ADC->ADCR |= (1 << 5) //sample on AD0.5 (pin 20)
                | (1 << 0) //sample on AD0.0 (pin 0)
               | (1 << 16) //BURST mode (repeated conversion)
                | (1 << 21);//operate ADC
```

c) selectSpeaker(input)

This section documents the code we used to set the enable signal that would go to each speaker. We first define 6 pins as output pins using the `DigitalOut` library: (`botL`, `botM`, `botR`, `midL`, `midR`, `top`). These correspond to the speaker at the position: (bottom left, etc).

The angular input, after filtering, is a D.C. signal with varying amplitude. We thus just split up the sampled input into 6 different channels, and enable set the corresponding pin:

```
if (input < threshold) {
   botL = 1; botM = 0; botR = 0; midL = 0; midR = 0; top = 0;
}
```

This is repeated for each channel, with different thresholds, and setting different pins.

d) main()

In our while loop in main, we read the value the of the sampled input and call `selectSpeaker()`. Then we poll for the interrupt bit that is triggered whenever the PWM resets. We then set the next duty cycle according to this value.

```
while(1){
          int pos = (LPC_ADC->ADDR0 >> 4) & 0xFFF;
          selectSpeaker(pos);
             if ((LPC_PWM1->IR >> 0) & 1) {//check interrupt bit for MR0
              int val = (LPC_ADC->ADDR5 >> 4) & 0xFFF; //sampled on pin20
              LPC_PWM1->MR2 = val >> 1; //div by 2 to fit into 0 - 2048
              LPC_PWM1->LER = 1<<2;//set Latch register to enable the changes
`             LPC_PWM1->IR |= 1<<0;//clear interrupt bit
         }
}
```

Amplification

The output of the mbed is a 0-3.3V signal. For an audio application, we need to amplify the power the decibel level is a metric of the signal power. Hence we chose the following power op amp TLE 2301 that had the following specifications:

a) High voltage limits ([-22, +22] V)
b) High current output (1 A)
c) High slew rate (12 V/micros)

We need a high slew rate since we are amplifying moderately high frequencies (40 kHz). The slew rate is the maximum rate of change of voltage the op amp is capable of, and we calculate a minimum slew rate as follows:

$$SR_{min} = \frac{\triangle V_{max}}{\triangle t_{max}}$$
$$\triangle\, t_{max} = T/10 = \frac{1}{40000 \times 10} = 2.5\mu\text{s}$$
$$\triangle\, V_{max} = 20\text{V}$$
$$SR_{min} = 8\text{V}/\mu\text{s}$$

We thus used a simple comparator circuit as shown in figure 6 that amplifies the $0 - 3.3$ V input to a ±Vcc output. We send an inverted signal to the other terminal of the speaker (as opposed to grounding it) to increase the current going through the speaker at any point in time. Each of the 6 arrays has its own amplifier. When the amplifier was powered with ±Vcc = ±15V, the current was about 0.3 A, giving a power consumption of 9 W.

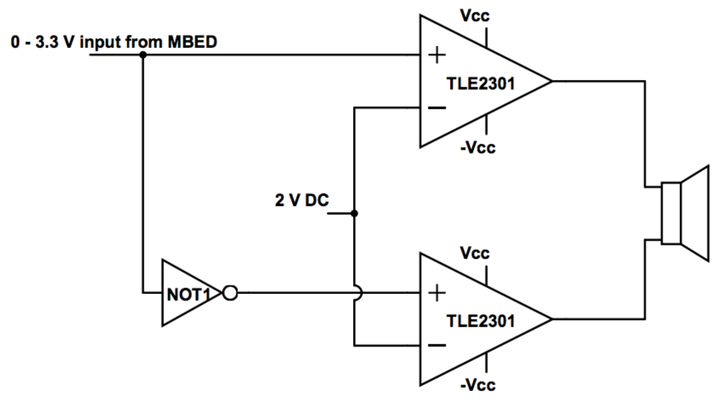


Figure 6. High frequency power op amp amplification circuit.

The circuit we wired up on a breadboard was very messy and complicated, so we had an amplifier dedicated PCB fabricated to make life much easier. Figure 7 shows the messy breadboard, PCB layout, and the fabricated PCB.

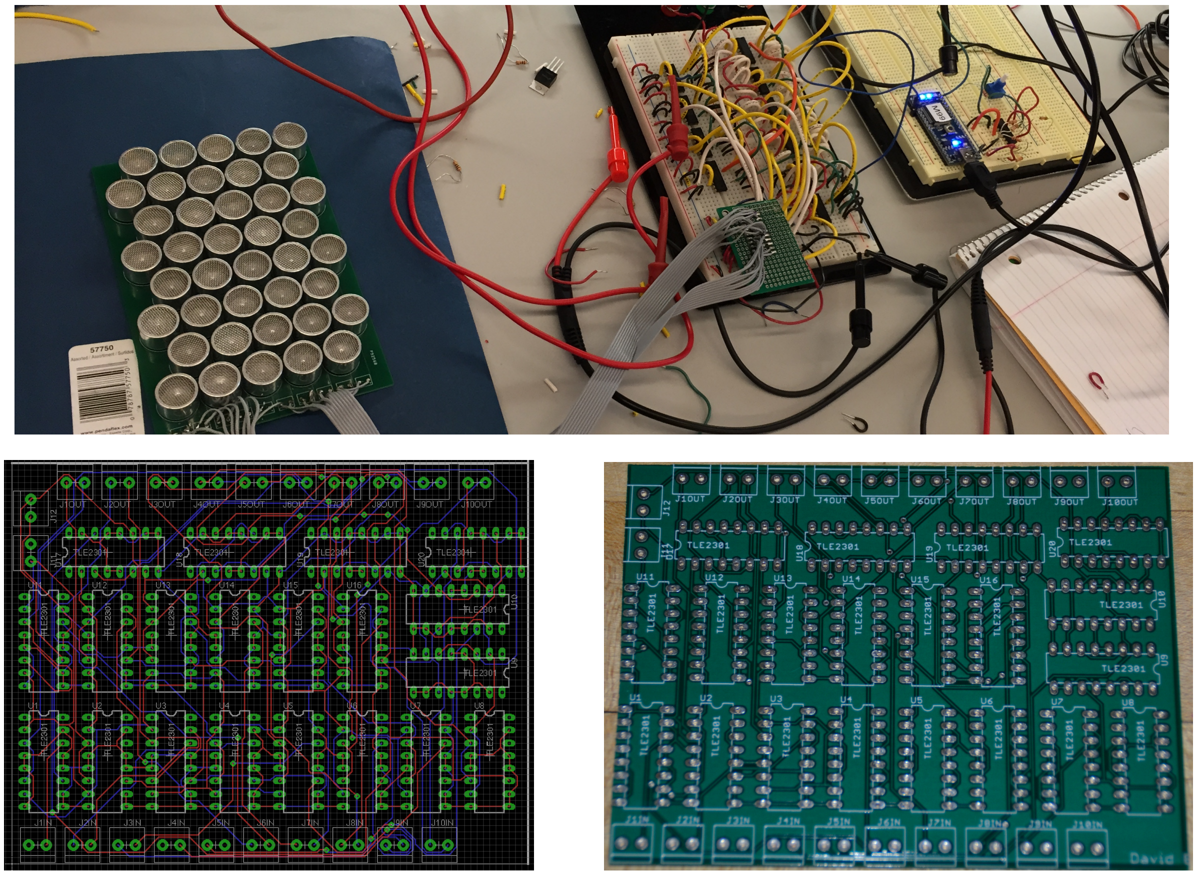


Figure 7. From hairy breadboard to clean PCB.

One issue we ran into was that the ultrasonic transducers introduced a lot of noise into signals. This affected how it sounded. This could because the impedance of the speaker brought in a lot of noisy feedback. We found that connecting shunt capacitors to the D.C. lines removed some of the noise, by shorting the high frequency A.C. noise to ground, as in figure 8.

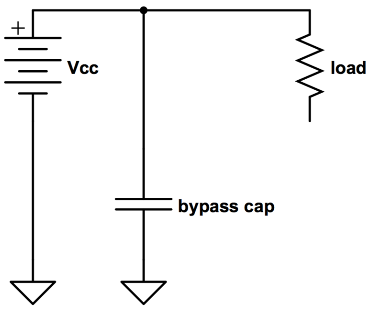


Figure 8. Shunt capacitor circuit for noise reduction.

Hardware

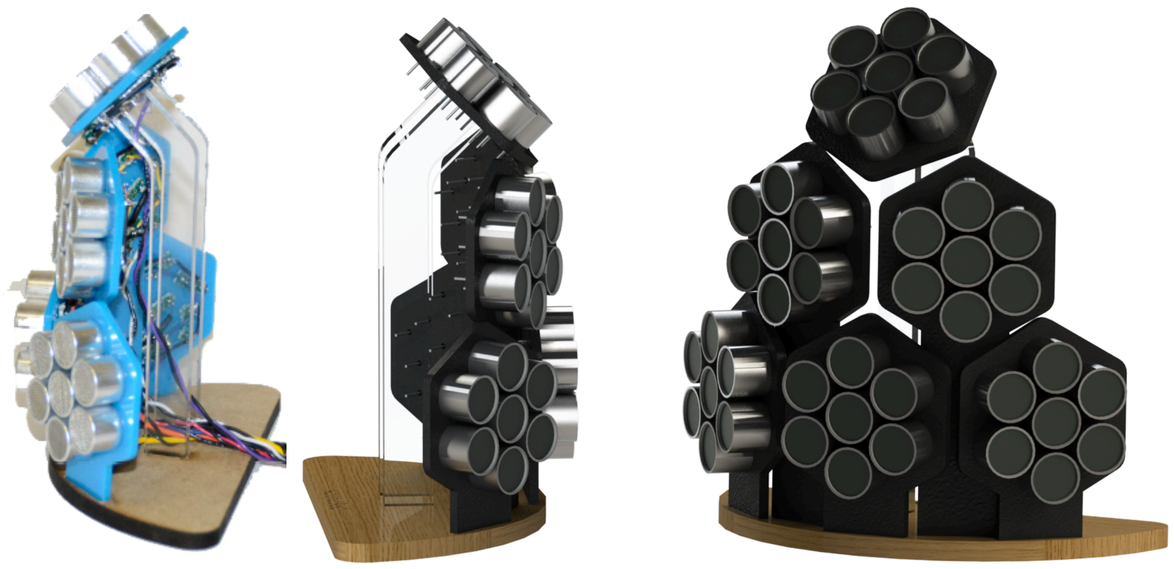


Figure 9. Speaker design.

Speaker

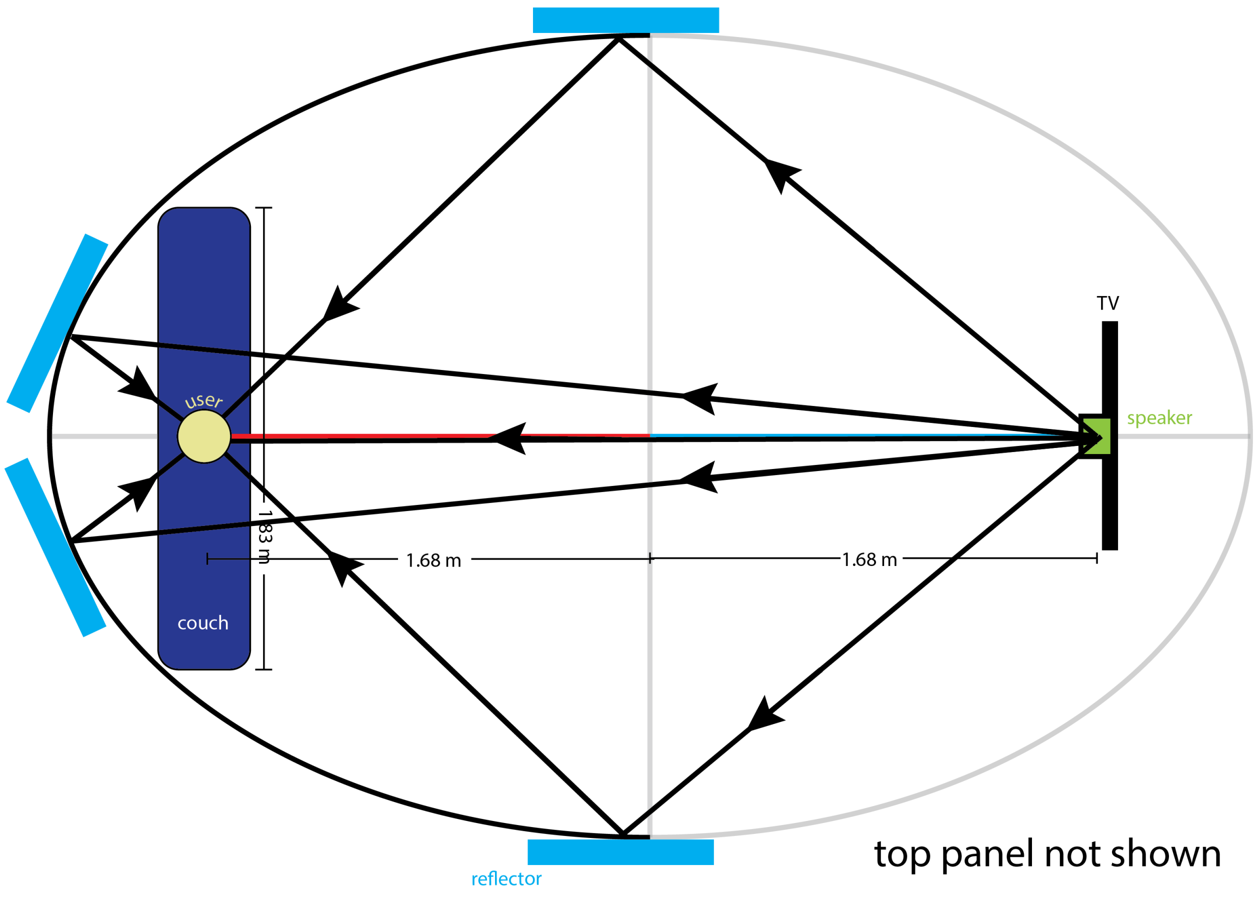
Our speaker design consists of six modules/arrays each containing 7 ultrasonic transducers. The transducers are hexagonally-close-packed to minimize the space between transducers, which is important for demodulation in the air. All parts are made of laser cut acrylic and MDF. The first page of this report shows the final sound360 system from a trimetric perspective. Figure ## shows the a couple of renderings of the CAD model created in SolidWorks and another photo of the completed design. With this design, we can get one module (or a combination of modules) to play sound at a time. The top module is for bouncing sound off the ceiling, while the reset are all in plane with the viewer.

Reflector Configuration

The beams of sound need to be reflected at the user from different angles in order to make the sound feel like it is coming from all around. We harness the geometric properties of the ellipse to keep our reflector design simple and minimally invasive. The speaker is positioned above the television (or other viewing screen) at one focus of the ellipse. The user/viewer is stationed at the other focus of the ellipse as shown in the figure ##. The modules are positioned at angles of 0°, ±10°, and ±35° from the horizontal axis. There is also a module that points up at 45° to bounce sound off the ceiling (or an upper reflector). The dimensions in figure ## are for the setup in XLab and can easily be adjusted to fit any home entertainment system. The eccentricity of the ellipse is independent of the ellipse's reflective property.

We found that the sound beams reflect well off of most solid surfaces, and so for the purpose of our demonstrations, we used 30-by-40" pieces of readily available poster board which were positioned at their correct locations tangential to an imaginary ellipse, as opposed to constructing a single continuous elliptical reflector. For our quick demonstrations in the Detkin Lab, we decided to just mark on the floor where and at what orientation these reflectors should be positioned. We then commandeered classmates to act as stands and had them hold the reflectors in place while a user was seated centrally. With this setup, we are able to simulate sound from 6 locations: left/right behind, left/right ahead, straight ahead, and above.

Figure 10. Elliptical geometry.



Video Demonstration

For our class demonstration, we synced up our device with the lobby scene from *The Matrix*. This scene features lots of gunfire and combat and thus works well with sound360. For intense action clips where shots are being fired from all around, we have the sound chaotically switch between random directions at about 2 Hz. Because of time constraints, we manually formatted the left audio channel to align with the visuals using Adobe Premiere. A better solution would be to just decode surround sound 5.1 audio and automatically choose which direction the sound should emit from. Our angular input encoding is very simple. In the left channel, we play a 2 kHz sine wave. We set the amplitude of this 'tone' to six different 'volumes', each corresponding to a different speaker module. Figure ## shows what this formatting looks like in Premiere.

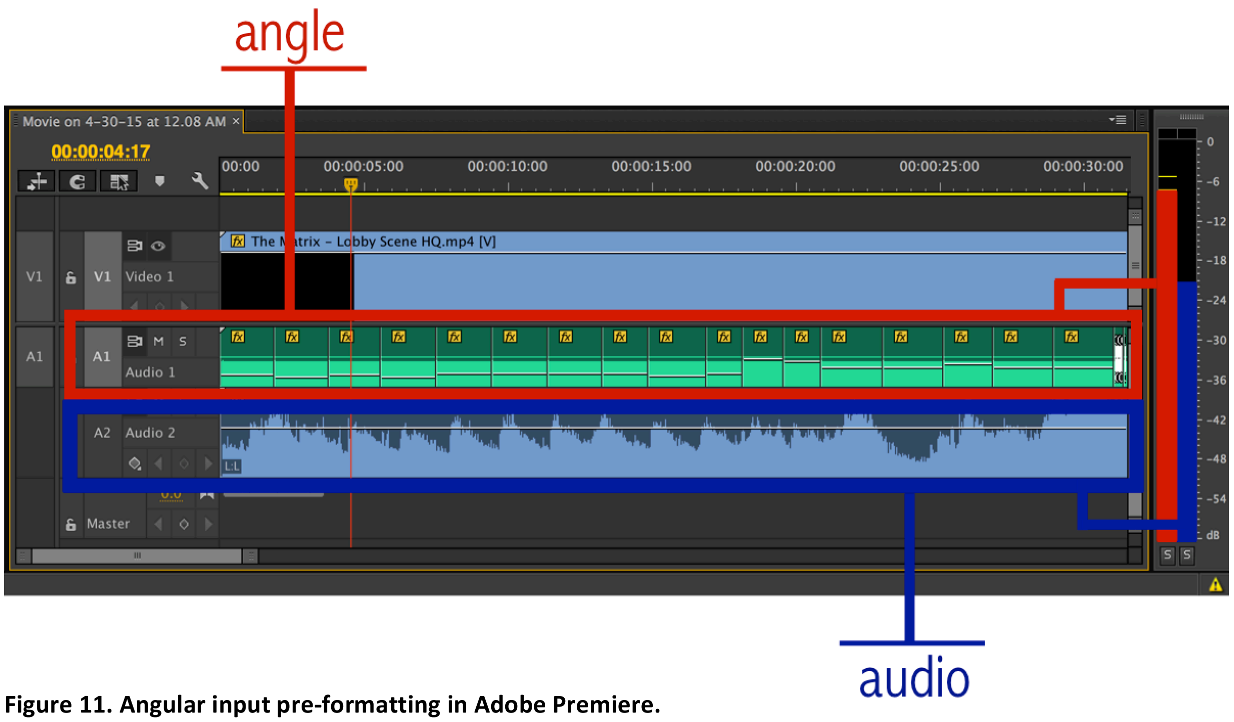


Figure 11. Angular input pre-formatting in Adobe Premiere.

Conclusion

sound360 was a major success, but there is still plenty of room for improvement. A great feature in our design is the modularity of the system. More modules can easily be added with minimal effort to get sound from even more directions. To make the sound louder, each module might need to contain more transducers, although we believe a better op amp could also supply more current and increase the volume. The size of the transducers themselves also affects the audio quality—higher quality audio might be obtainable with smaller transducers because they can be packed closer together.

The reflector configuration is currently rather rudimentary and could drastically be updated. First, more materials could be tested to see which is the *best* at reflecting the sound beams. The reflectors could then be mounted in a stable environment (eg mounted on 80/20 pillars in XLab). Alternatively, a continuous elliptical (or even ellipsoidal) reflector could be custom constructed for a home theatre. Another idea is that the system could determine the physical layout of the room with a Kinect and automatically adjust the beam directions accordingly, perhaps with phasing.

Instead of manually formatting the angular input, we could take advantage of 5.1 surround sound which already has the audio formatted for 5 directions plus a sub. A small change in our setup would be necessary to play sound simultaneously from multiple speakers, and we would need to use a power supply that can provide more power than the ones in Detkin, but it is definitely achievable. sound360 could also be used in applications other than entertainment, such as advertising. A beam of sound could continuously track an unsuspecting consumer and deliver individualized commercial messages. The possibilities are endless.