CSCI1600 Lab 3: Sound Part I

September 25, 2019

1 Objectives

By the end of this lab, you will:

- Connect a speaker and play a tone
- Use the speaker to play a simple melody
- Implement a single amplifier

Materials: We will be providing the parts necessary for this lab (other than the Arduino and breadboard).

2 Background: Sound

Consider Wikipedia’s definition of sound: “a vibration that propagates as a typically audible mechanical wave of pressure and displacement”

A speaker produces sound by moving a cone back and forth to generate these pressure waves.

Figure 1: Sound wave. Source: physicsclassroom.com
A speaker is driven using electromagnetism: applying current in one direction pulls the speaker cone in one direction and applying current in the opposite direction pushed the speaker cone in that opposite direction. Therefore, applying an oscillating current will cause the speaker cone to move back and forth repeatedly, creating an audible sound. Sound waves can measured by frequency, which is the number of oscillations per second, referred to as Hertz (Hz).

In this lab, we will use our Arduino to generate simple sounds consisting of a single frequency. We will expand on this to show how more complex sounds can be reconstructed via sampling, which you will use in your next project.

The frequencies corresponding to common notes can be found here:

http://www.phy.mtu.edu/~suits/notefreqs.html

3 Connecting the Speaker

Now we are going to wire up our speaker to our Arduino’s PWM port to apply the concepts above to make sound! To connect the speaker, you will need:

1. An 100 Ω resistor (You can use a larger resistor, but it will decrease the volume)
2. An 8Ω speaker

Connect your circuit to match this diagram:

![Basic Speaker Circuit](image)

Figure 2: Basic Speaker Circuit
To connect the speaker, use the two pins that are located closest to each other as the terminals—you can push these into your breadboard. Figure 4 shows an example for how to connect the speaker to a breadboard.

Task: Connect your speaker to your Arduino to match the configuration in the figure.

4 Playing a simple melody

We can create simple sounds with our Arduino by generating PWM signals (ie, square waves) at a specific frequency. We can use this to play a simple melody by playing “notes” at a given frequency for a certain amount of time.

Task: Use the tone() function to make your speaker play a simple melody. You can use the tone function to specify the pitch and duration of each note.

Note: Instead of playing random noise, here are the music notes to their corresponding frequencies.

5 A simple amplifier

Our speaker is rather quiet. Part of this is unavoidable: our speaker is only rated for 0.2W, so it cannot produce a very powerful signal. However, we can provide some additional power and improve the quality using a single transistor—an example circuit is shown in Figure 5. The transistor basically operates as a voltage-controlled switch, allowing a low-power source to drive a high-power load. The transistor diagram is shown in Figure 6. The input signal from the Arduino controls this “switch” to provide additional power to the speaker supplied by the Arduino board’s voltage regulator, rather than powering it directly from the PWM output pin.

Warning: Be sure that you connect the capacitor C1 correctly! Large capacitors are polar components and must be connected in the correct direction. The negative end of the capacitor is marked with a grey stripe on the body of the capacitor and should be connected to the speaker. Connecting the capacitor incorrectly can cause explosive results!

Note: The transistor used in this schematic has multiple variations with different pin configurations. To ensure you connect it correctly, look up the datasheet for the part number of the transistor you are using to find its pin diagram.

Figure 5: Sound single transistor amplifier schematic
6 Reading: Sampling: Producing more complex sounds

To produce arbitrary sound waves, we would need to be able to output any analog voltage on a spectrum (0 to 5V, for example) in order to perfectly match our wave. Our Arduino, however, can only output digital signals: HIGH (5V) or LOW (0V). Calling the `tone()` function generates a PWM waveform on a particular pin (ie, a square wave) at a particular frequency. This works well for playing simple sounds consisting of single-frequency tones, but we need to go a step further to produce more complex signals.

To approximate an arbitrary, continuous signal, digital systems represent these signals as a discrete series of samples, which are scalar values that represent the magnitude of the signal at a certain time index. An example of this is shown in Figure 7, with each $S_i$ representing a discrete sample of the analog signal $S(t)$.

![Figure 7: Sampling. Source: wikipedia.org](image)
Note that each sample is recorded at a periodic rate, $T$, the inverse of which is called the sampling frequency or sample rate. A full discussion of sampling theory is far beyond the scope of this lab. However: for the purpose of reproducing a waveform, this means that we need to output a new sample at every time interval $T$ in order to accurately reconstruct the signal—a hard real-time operation.

Furthermore, it makes intuitive sense that we cannot reproduce waveforms of a frequency higher than our sampling frequency. In fact, to perfectly reconstruct a waveform without losing information, the maximum frequency we can reproduce is equal to half the sampling rate—a fundamental theorem in sampling theory by Claude Shannon and Harry Nyquist. You can read more about this here: https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem
## 7 Grading Rubric

<table>
<thead>
<tr>
<th>Task</th>
<th>Total Points</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speaker circuit on breadboard</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Playing simple melody with <code>tone()</code></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Blinking LED using interrupts</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Playing melody using interrupts</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>(Optional) Simple amplifier circuit</td>
<td>(10)</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>30</strong></td>
<td></td>
</tr>
</tbody>
</table>