Exploring Sound and Music with Arduinos

Zoheyr Doctor
Email: zdoctor@uchicago.edu, zoheyr@gmail.com
Resources: https://github.com/zodoctor/SpaceExplorers/tree/master/NSTA2018
Websites: http://kicp.uchicago.edu/education/explorers, zodoctor.github.io
Twitter: @almostdrdoctor

March 17, 2018

Abstract

Welcome! In this hands-on session, we will look at three activities which middle or high schoolers can do with the essential do-it-yourself microprocessor Arduino.

The three activities are:

1. Flash an LED and play a tone from headphones with the Arduino
2. "Light Theremin": Control the pitch played by the headphones using a light sensor
3. "Electronic Trumpet": Three buttons can be pressed to play different notes in a scale

Learning objectives for these activities include:

- Build circuits on breadboards by reading circuit diagrams
- Modify code snippets to e.g. change the tone played from headphones
- Map voltage values read in by sensors to changes in the system output
- Create a new electronic instrument or modify an existing one using knowledge accrued in the activities
- Calculate frequency from the period

1 Introduction

1.1 Acknowledgements and References

Much of the following has been adapted from Arduino DIY activities or Arduino example codes I've seen online, so I am certainly not the first person to do these kinds of activities! This is one of the really nice things about the open-source environment for Arduinos – there are multitudes of pre-existing projects online that you can use as a guide in developing your lessons. I highly recommend checking out the website arduino.cc for general Arduino code and examples, and sparkfun.com for parts.

1.2 SAFETY

Arduinos can only supply 5V, making them harmless in terms of shock hazard. Nevertheless it is best to use safe practices when building these electronics, especially because other electronics students may encounter will have higher voltages and/or source more current. Some good rules of thumb which will promote safety and the health of your electronics:

- Always unplug the Arduino/circuit from power when modifying the circuit.
- Double check that you are using the right circuit element before powering – students can easily fry resistors or Arduinos by choosing the wrong resistor value.
- Do not switch in higher voltage power sources ad-hoc.
- Be careful with ends of wires as they can be sharp.

1.3 Pre-requisite Knowledge

I highly recommend that students have some pre-requisite knowledge about sound, waves, and electronics before embarking on the activities below. Alternatively, parts of the activities could be used in teaching basics of sound, waves, and electronics. Below, I summarize key concepts for these electronics activities.

1.3.1 Sound

Sound travels through air via collisions of neighboring air particles. A sound wave is the back and forth sloshing of particles due to these collisions, which can be characterized by three quantities:

- *Amplitude*: The distance moved by particles as they slosh back and forth
- *Frequency*: How many times in a time interval (e.g. a second) the particles slosh back and forth
- *Timing/Phase*: When the particles started sloshing, and in which direction.

To create a desired sound wave, one must force particles to move back and forth with the right amplitude, frequency, and phase. You can also create multiple sounds at once (e.g. a chord), by simply adding the forces needed for all the sound waves you want to combine. To actually create those desired forces, we often use speakers. A speaker is essentially a paper cone, which vibrates with the right amplitude, frequency, and phase to perturb nearby air particles, which bump into their neighbors, which bump into their neighbors, and so on to transmit a sound to your ear. A really nice way to see the movement of a speaker is to play a pure tone from the speaker while shining a strobe light at it.

1.3.2 Electronics

How is the speaker cone actually moved back and forth in the right way and so quickly? The back end of the cone is connected to a permanent magnet, which is nearby an *electromagnet*. One can change the strength and polarity of the electromagnet by changing the amount and direction, respectively, of electrical current going through it. This changing magnetism of the electromagnet with time will cause the permanent magnet – and hence the speaker cone – to move back and forth. How a smartphone headphone jack, for example, is able to change the direction and strength of that electrical current which goes to the speaker is beyond the scope of these activities, as it becomes complicated quickly.

Electrical current is the flow of electrons in a circuit. One can conceptualize these flows as analogous to flows of water in pipes. Water in a pipe wants to go from high energy to low energy – e.g. water will go downwards in a vertical pipe if there are no forces other than gravity. *Voltage* V (typically with units of Volts V) is the electrical analog of the energy or height in the previous example. The *electrical current* I (units of Amps A) is just like water current: it is how much electrical charge is passing through a wire in some time interval. *Resistance* R (units of Ohms Ω) just tells us how much current flows between two points if we know the difference in voltage ΔV between those two points:

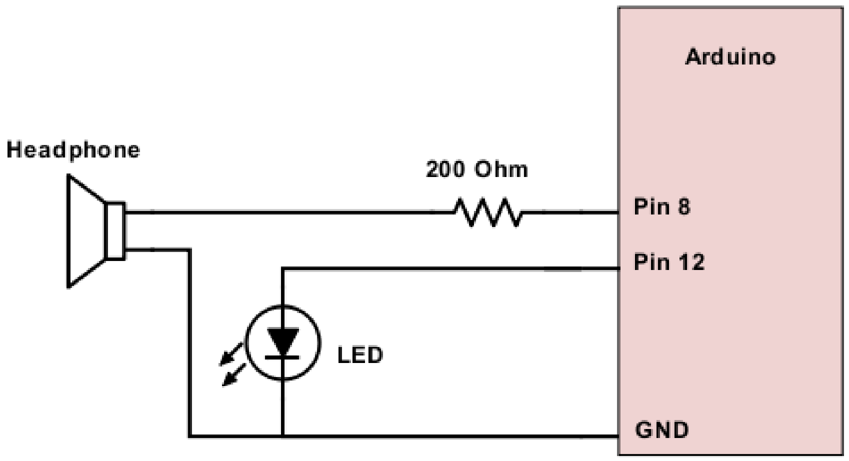
$$\Delta V = IR$$

The electrical resistance, analogously to resistance to water flow, is dependent on the 'pipe' or wire it is flowing through. In water pipes, one might put in valves to slow down the flow of water to the desired level. Resistors in electronics act similarly to impede the flow of electrons. The flow can *do work*, just like flowing water can move a waterwheel.

Lastly, one must know how to put together electrical circuits and read circuit diagrams. The activities below can be used to get some basic experience on building the circuits, but there are some things students should know before diving in:

- A *breadboard* is a board with holes, some of which are connected via metal which allows current flow. Wires can be slotted into certain holes in order to connect wires together. This allows one to prototype quickly without having to add in fancy connectors.
- A *circuit diagram* is picture showing how each circuit element is connected to other elements. The connecting elements in the diagram are simply *connections*, **not the spatial layout of the circuit**. You can think of the diagram as analogous to a diagram you would make to show someone how to set up their home entertainment system. You wouldn't show them where to put the wires in their house since each person's house is different. You'd just say which ports of which devices need to be connected with cables.

Figure 1: Circuit diagram for flashing an LED and playing a tone



With all this said, there are certainly other bits and pieces about electronics which need to be taught or motivated as one goes along (e.g. GROUND is the place where we all agree the voltage is zero, like we can all agree that sea level is where we define altitude to be zero). I leave these bits and pieces to you to tailor to your own teaching needs.

2 Flashing LED / Playing a Tone

This activity is a good warm-up for the others, because it requires few circuit elements and has a straightforward design. Figure 2 shows the circuit diagram for the setup. With this built, we can now take a look at the code, which you can also find on my github page in the .ino file `BlinkLEDandPlayTone.ino`:

```
/* This script blinks an LED and
/* plays a tone, both at the same
/* frequency.  The frequency is specified
 * below.
 */
int ledPin = 12;      // select the pin for the LED
int speakerPin = 8; // select pin for speaker output
double frequency = 1; // declare variable, Hz
unsigned int periodMilliseconds = 0; // declare variable, milliseconds

void setup() {
  // declare the ledPin and speakerPin as OUTPUTs:
  pinMode(ledPin, OUTPUT);
  pinMode(speakerPin, OUTPUT);
  // open the serial port at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // The snippet below is looped.  In each
  // loop, the speakerPin and ledPin are told to go to
  // HIGH and then LOW voltage.  They wait
  // at HIGH or LOW for time given by
  // periodMicroseconds.

  // STUDENTS:
  periodMilliseconds = 1000/frequency;
  // print out the periodMilliseconds value
  // to make sure things are working.  Click
  // the magnifying glass button in the top
  // right to see the serial output.
  Serial.print(periodMilliseconds);
  Serial.print("\n");


  // tell the speaker and LED pins to go HIGH
  digitalWrite(speakerPin, HIGH);
  digitalWrite(ledPin, HIGH);
  // Wait for periodMilliseconds/2
  // STUDENTS
  delay(periodMilliseconds/2);

  // tell the speaker and LED pins to go LOW
  digitalWrite(speakerPin, LOW);
  digitalWrite(ledPin, LOW);
  // Wait for periodMilliseconds/2
  // STUDENTS
  delay(periodMilliseconds/2);
}
```

Lines right below where I have commented `STUDENTS` are lines that the students can fill in, or lines where the students can fill in the arguments to function calls like `delay(argument)`. With the code completed, it can be uploaded to the Arduino with the right arrow button at the top of the Arduino GUI window.

2.1 Activities

Below are some example activities, which are just the tip of the iceberg.

1. Set the frequency value to 1 Hz and upload to the board. What do you see from the LED? What do you hear from the headphones? Do these make sense given the 1 Hz frequency?
2. Now do the same with `frequency=100`. What do you see and hear? Does the LED appear to be flashing? Why or why not?
3. Try more frequency values between 0.1 Hz and 2000 Hz. From these trials, what is the relationship between frequency and the pitch you hear? Is a higher frequency a higher pitch or a lower pitch?
4. *Optional*: Can you add in another LED on, say, Pin 9 which will blink off when the other LED is on and blink on when the other LED is off?

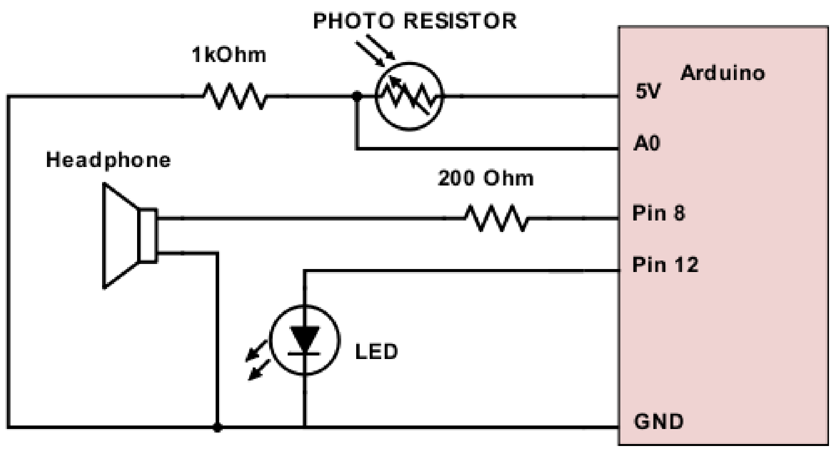
3 Flashing LED / Playing Tone Based on Photoresistor: The "Light Theremin"

One irritating aspect of the design in Section 2 was the fact that you had to re-upload the code every time you wanted to change the frequency. What if instead we could change the frequency on the fly? One way to do this would be to have some sort of knob that we could turn that the Arduino could read and then use to control the frequency. Rather than using a knob, let's use a photoresistor, a little photocell which changes resistance based on how much light is incident on it. You can move your hand near and far from the sensor to change the pitch, just like a theremin! The circuit diagram for it is shown in Figure 3. And here's the accompanying code, also with

3.1 Activities

- Click the magnifying glass in the top right of the Arduino window to see the values that are being printed out by the code. These values correspond to the voltage levels on Pin A0. They are not in volts, but you can still see that they change depending on how much light is on the photocell. How can you change the code to make the range of frequencies played by the light theremin different?
- *Optional*: Can you add another photocell to allow the LED and speaker to have different frequencies?

Figure 2: Circuit diagram for flashing an LED and playing a tone with frequency depending on the light on a photoresistor



4 "The Electronic Trumpet"

Lastly, we have the electronic trumpet. There are three buttons, and pressing different combinations of the buttons plays different notes, just like a trumpet. The circuit diagram to attach one button is is shown in Figure 4.1. The code is below and is in the file `ThreePushbuttonScale.ino`. It must be accompanied by `pitches.h`, because that file defines all the notes in the scales.

```
/* Code for the "electronic trumpet.
 *  Plays different tones from a speaker
 *  depending on which combinations of
 *  buttons are pressed.  The version here
 *  plays notes in the C major scale, but
 *  that can easily be modified.  This
 *  script depends on pitches.h,
 *  which is a file that
 *  defines the frequency of each note.
 */

#include "pitches.h"

int ledPin = 13; // choose the pin for the LED
int inPinA = 5;   // choose the input pin (for pushbutton A)
int inPinB = 6;   // choose the input pin (for pushbutton B)
int inPinC = 7;   // choose the input pin (for pushbutton C)
int speakerPin = 8;
int valA = 0;     // variable for reading the pin status
int valB = 0;     // variable for reading the pin status
int valC = 0;     // variable for reading the pin status

void setup() {
  pinMode(speakerPin,OUTPUT); // declare speaker as output
  pinMode(ledPin, OUTPUT);  // declare LED as output
  pinMode(inPinA, INPUT);    // declare pushbutton as input
  pinMode(inPinB, INPUT);    // declare pushbutton as input
  pinMode(inPinC, INPUT);    // declare pushbutton as input
}

void loop(){
  valA = digitalRead(inPinA);  // read input value on Yellow button
  valB = digitalRead(inPinB);  // read input value on Yellow button
  valC = digitalRead(inPinC);  // read input value on Yellow button
  if ((valA == HIGH) and (valB == HIGH) and (valC == HIGH)){
  // check if the input is HIGH (button released)
    noTone(speakerPin);  // turn sound off
  }
  else if ((valA == HIGH) and (valB == HIGH) and (valC == LOW)) // C pressed
  {
    // Here the 'tone' function is used.  The inputs to
    // tone are the pin number on which you want to output
    // the sound, the frequency of the note, and the length
    // of time in milliseconds to play the tone, respectively
    tone(speakerPin,NOTE_C5,20);  // turn sound on
  }
  else if ((valA == HIGH) and (valB == LOW) and (valC == HIGH)) // B pressed
  {
    tone(speakerPin,NOTE_D5,20);  // turn sound on
  }
  // STUDENTS fill in the remaining if/thens
  else if ((valA == LOW) and (valB == HIGH) and (valC == HIGH)) // A pressed
  {
    tone(speakerPin,NOTE_E5,20);  // turn sound on
  }
  else if ((valA == HIGH) and (valB == LOW) and (valC == LOW)) // B and C pressed
  {
    tone(speakerPin,NOTE_F5,20);  // turn sound on
  }
  else if ((valA == LOW) and (valB == LOW) and (valC == HIGH)) // A and B pressed
  {
    tone(speakerPin,NOTE_G5,20);  // turn sound on
  }
  else if ((valA == LOW) and (valB == HIGH) and (valC == LOW)) // A and C pressed
  {
    tone(speakerPin,NOTE_A5,20);  // turn sound on
  }
  else if ((valA == LOW) and (valB == LOW) and (valC == LOW)) // all pressed
  {
    tone(speakerPin,NOTE_B5,20);  // turn sound on
  }
  else {
    noTone(speakerPin);  // turn sound off
  }
}
```

4.1 Activities

- How many distinct notes can you make with pressing combinations of 3 buttons? What about N buttons? (This last one is not easy)
- Can you add a fourth button to allow for more notes?
- Can you add one LED for each button that is bright whenever that button is pushed?

Figure 3: Circuit diagram for connecting one button

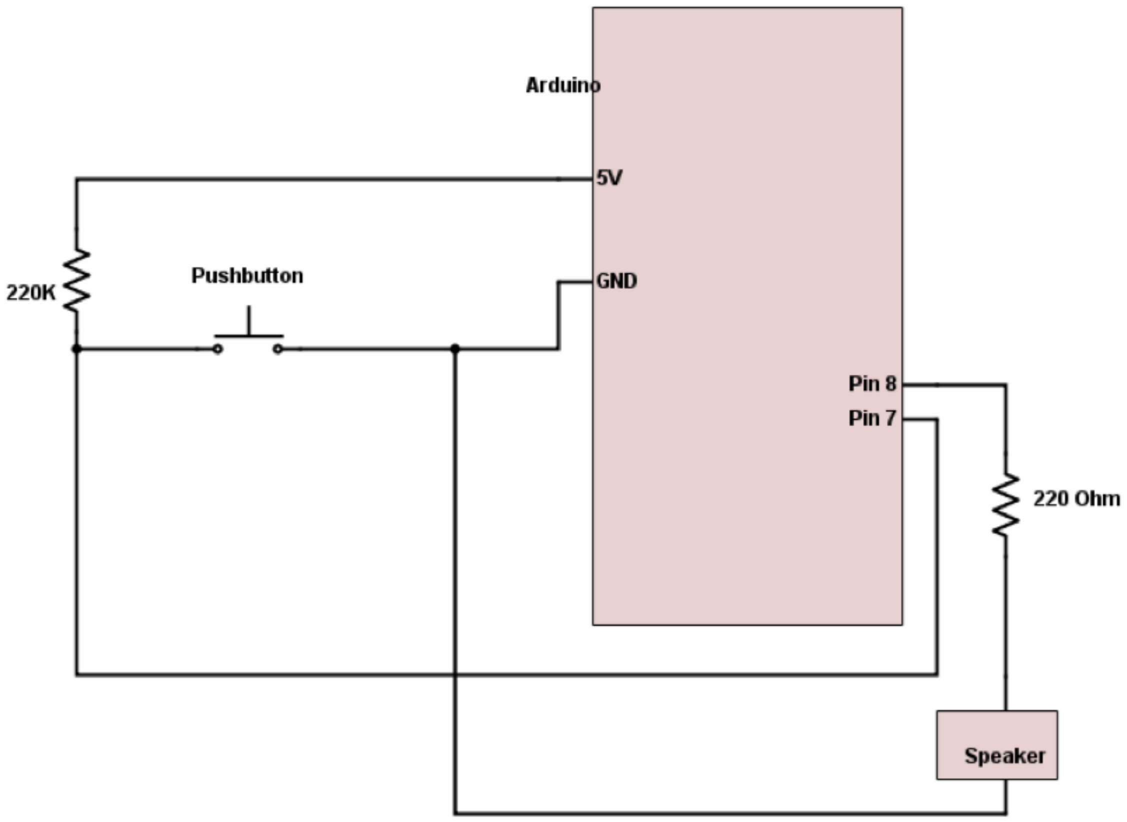
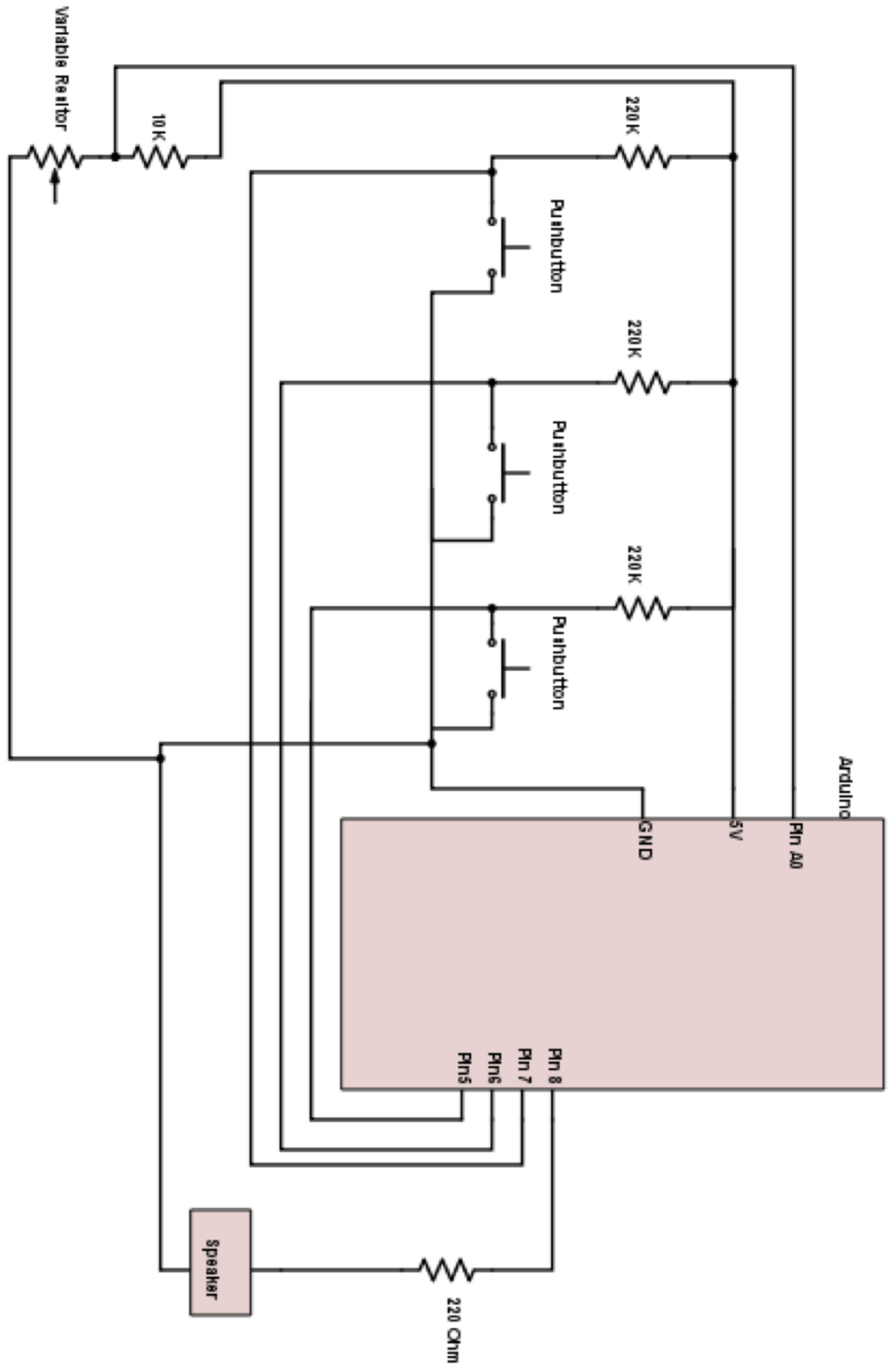


Figure 4: Circuit diagram for the electronic trumpet


Variable Resitor
10K
220K
220K
220K
Pushbutton
Pushbutton
Pushbutton
Arduino
GND
5V
Pin A0
Pin 8
Pin 7
Pin6
Pin5
Speaker
220 Ohm