PHYS360: Arduino Project - Sound Detector & Visualizer

Nathan Pierce

Concept:

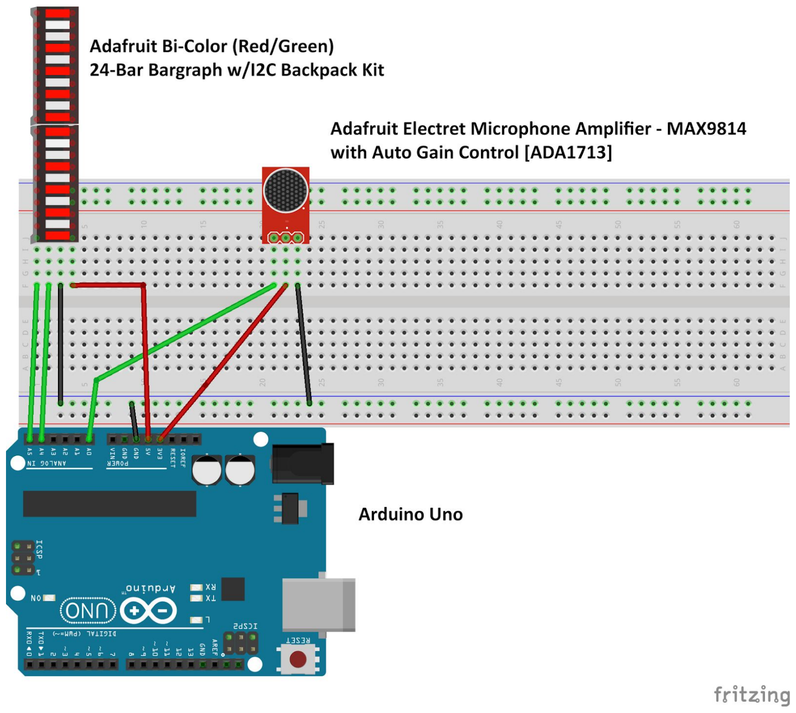
The idea behind the project is straightforward - use an Arduino board with the ATMEGA controller to serve as the platform for simple sound detection and visualization. The project is largely based on an Adafruit project[1]. A microphone detects ambient sound and converts it to an analog voltage, which the Arduino then reads and converts to an integer value that is then sent to bar graph to display some number of LEDs corresponding to the amplitude of the signal (i.e. the loudness of the sound).

Parts & Tools:

- Adafruit Bi-Color (Red/Green) 24-Bar Bargraph w/I2C Backpack Kit
- Adafruit Electret Microphone Amplifier - MAX9814 with Auto Gain Control [ADA1713]
- Arduino Uno
- Fluke 115 True RMS Voltmeter
- Soldering Iron
- Breadboard

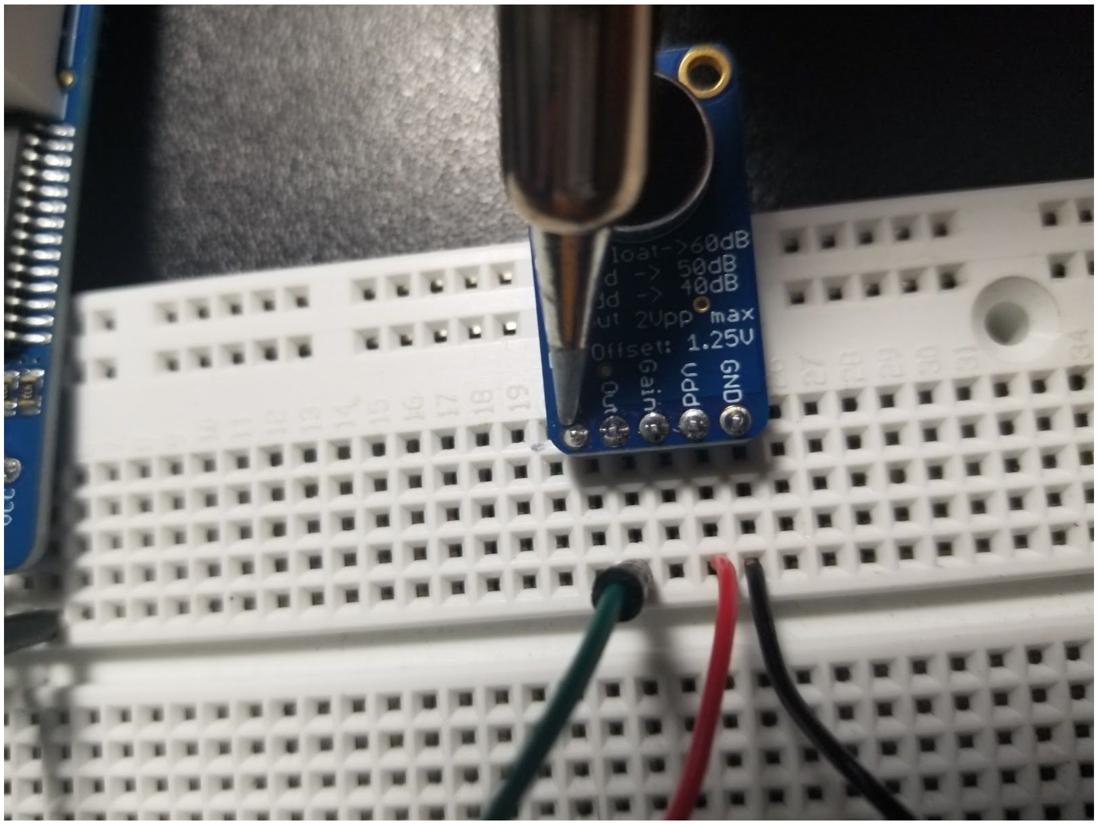
The Build Process:

Having acquired all the necessary components, I first set about soldering the header pins for the bar graph and microphone, then making all the necessary connections.


Adafruit Bi-Color (Red/Green)
24-Bar Bargraph w/I2C Backpack Kit
Adafruit Electret Microphone Amplifier - MAX9814
with Auto Gain Control [ADA1713]
Arduino Uno
fritzing

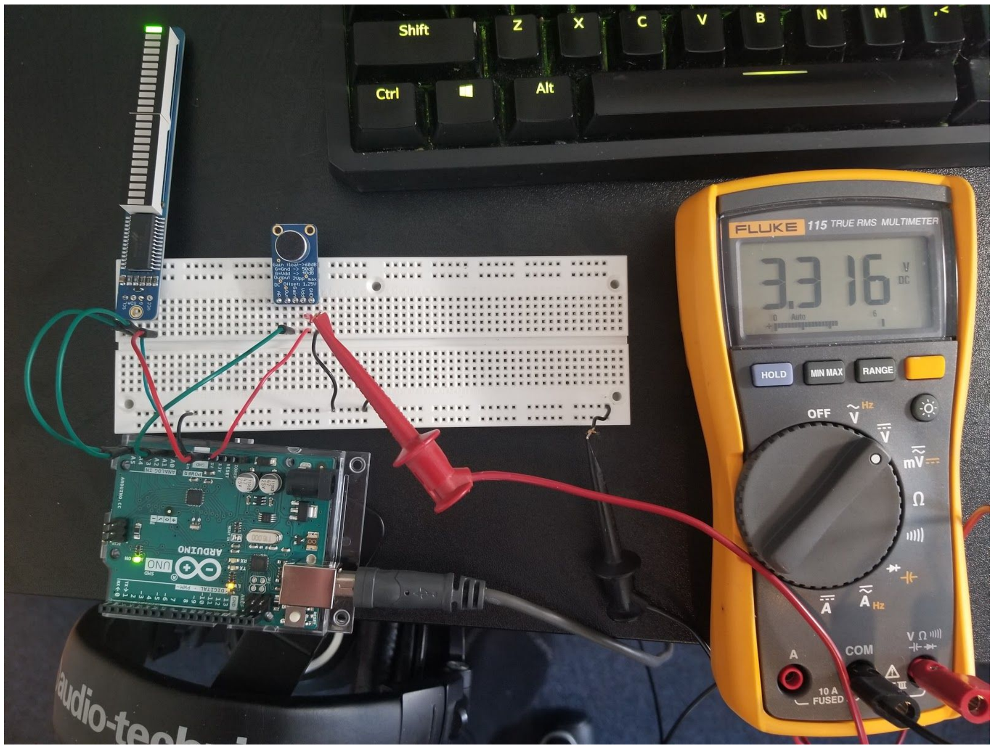
[1] Earl, Bill, and lady ada. "Adafruit Microphone Amplifier Breakout." *Adafruit Learning System*, Adafruit, learn.adafruit.com/adafruit-microphone-amplifier-breakout/measuring-sound-levels?view=all.

The green jumpers are for signals, the red for power, and the black for ground.



The microphone has a built in op amplifier with an automatic gain. The amp has a high gain - first, a fixed 12 dB gain, then a variable gain range from the automatic gain controller of 20 dB to 0 dB (depending on the output voltage), and finally selectable gain of 8, 18, or 28 dB. The cascade of the multistage amp results in a high gain - up to 60 dB. The data sheet is included here. I found the microphone to be quite sensitive, which helped to make the visualization more responsive.

I used the DMM to confirm supply voltages to the amp and the bargraph. After completing the hardware portion of the project, I moved on to coding the controller program.



First I downloaded the bargraph drivers, including 3 different libraries by ADAFruit - Adafruit GFX Library, Adafruit BusIO and LED Backpack Library. The Wire dependency is included with the Arduino installation.

```
#include <Wire.h>
#include "Adafruit_LEDBackpack.h"
#include "Adafruit_GFX.h"
```

I initialized some constants I would be needing later on for the logic, as Ill as created the bargraph object. The maxScale constant represents the number of LEDs on our bargraph, and will come in handy when we write the bar graph for loop.

```
Adafruit_24bargraph bar = Adafruit_24bargraph();

const int sampleWindow = 50; // Sample window width in mS (50 mS = 20Hz)
unsigned int sample;
const int maxScale = 24; // Number of LEDs on the bar graph
```

The setup() function is used to allow for us to monitor serial data transmission, which will come in handy for monitoring the real time output voltage of the amplifier/microphone. Here I also initialize some class variables of bargraph such as LED brightness.

```
void setup()
{
   Serial.begin(9600);

   bar.begin(0x70); // pass in the address
   bar.setBrightness(15);
   bar.blinkRate(0);}
```

Now it is time to begin using the data from the sample to determine peak-to-peak voltage as a function of sound pressure near the microphone. The Arduino-supplied millis() function allows us to start our sample window. It returns the time that has elapsed since the program started running. I use the startMillis variable as our start time. I also initialize the amplitude variable, peakToPeak, and the min/max signal variables.

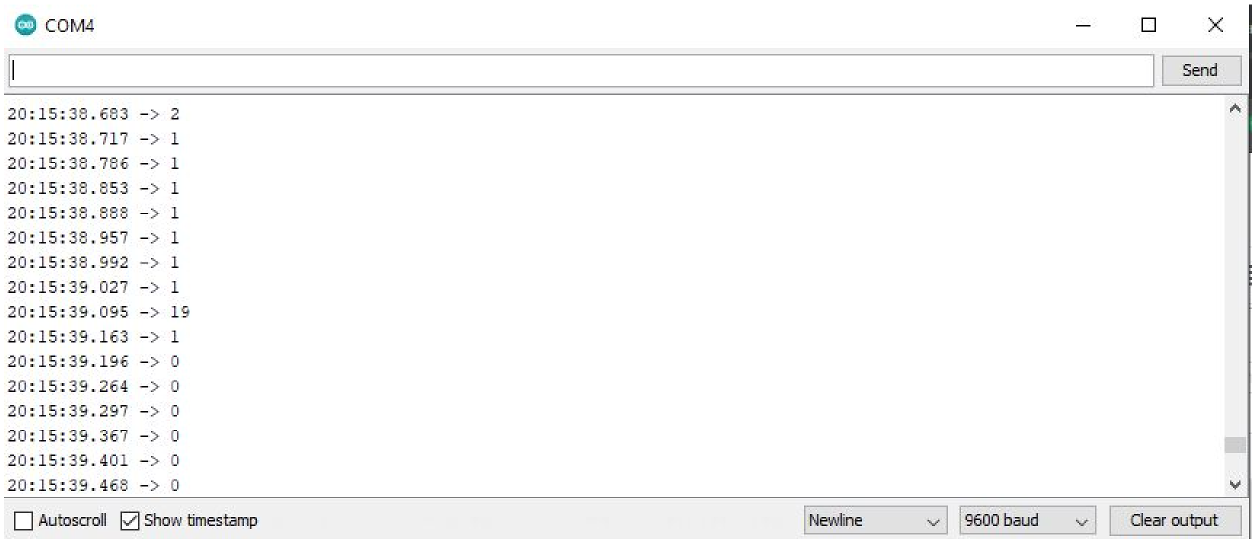
```
   unsigned long startMillis= millis();  // Start of sample window, t = 0
   unsigned int peakToPeak = 0;   // Amplitude measured as peak to peak
voltage, initialized at zero initially

   unsigned int signalMax = 0;
   unsigned int signalMin = 1024;
```

This while loop is where the voltage of the amplifiers output is read and stored. The while loop runs for the duration of the sample window - in this case, 50 ms. A voltage is measured 20 times every second. Then, it breaks the loop and continues on through the loop() function, to be run by the controller indefinitely. The output of the amplifier is stored in sample, which is assigned the value returned by the analagRead() function, which we pass the first analog input (analogRead(0)), since that is where the output from the amp is sent.

```
   while (millis() - startMillis < sampleWindow)
   {
      sample = analogRead(0); // reads analog voltage output to pin A0 on
arduino
```

```
        if (sample > signalMax)
        {
           signalMax = sample;  // save just the max levels
        }
        else if (sample < signalMin)
        {
           signalMin = sample;  // save just the min levels
        }
    }
```

Now that we have the maximum and minimum voltages, we can determine the "amplitude" of the sound - the peak-to-peak voltage difference of the signal measured. I now have the amplitude of the signal, and can now use it to drive the LEDs on the bar graph. To do this, I need to first convert the peakToPeak voltage to an integer to be used in the following logic that determines how many LEDs should turn on for a given amplitude. The map() function does just this! Note that displayPeak will range from 0 to 24. Below is a screenshot of the serial monitor reacting in real time to sound in the environment (in this case, a snap from my fingers).



```
20:15:38.683 -> 2
20:15:38.717 -> 1
20:15:38.786 -> 1
20:15:38.853 -> 1
20:15:38.888 -> 1
20:15:38.957 -> 1
20:15:38.992 -> 1
20:15:39.027 -> 1
20:15:39.095 -> 19
20:15:39.163 -> 1
20:15:39.196 -> 0
20:15:39.264 -> 0
20:15:39.297 -> 0
20:15:39.367 -> 0
20:15:39.401 -> 0
20:15:39.468 -> 0
```

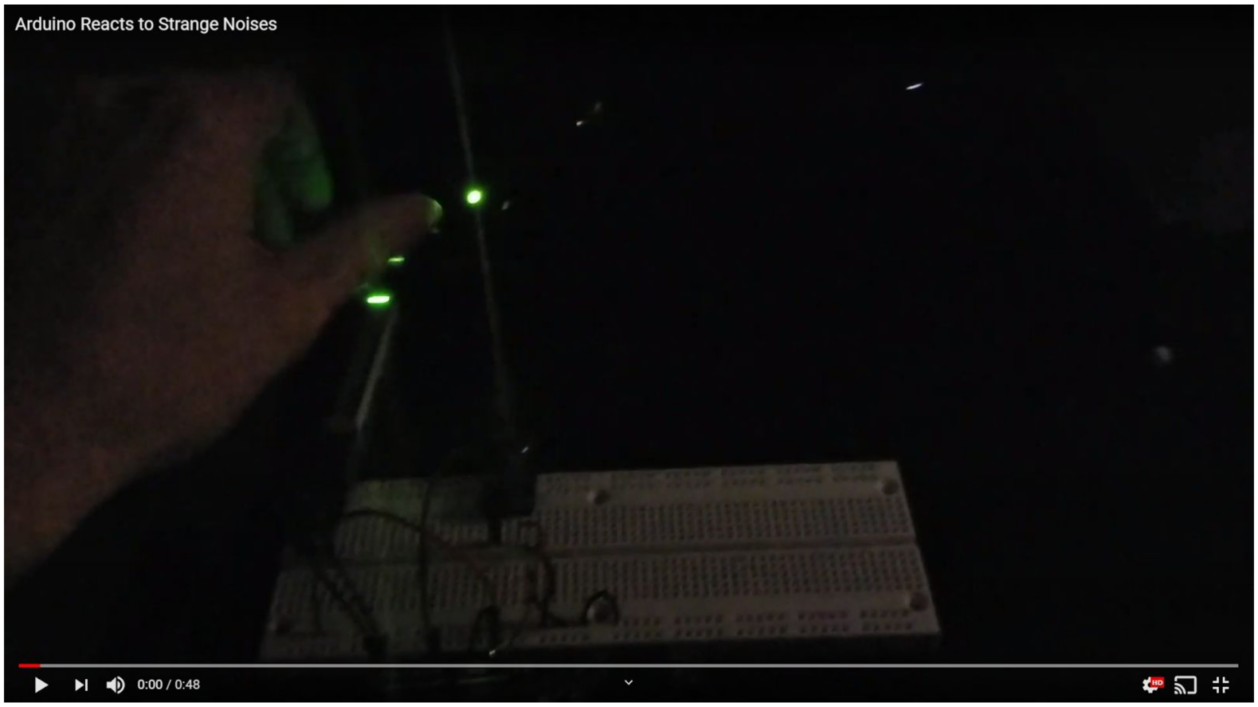
```
    peakToPeak = signalMax - signalMin;  // max - min = peak-peak amplitude
    int displayPeak = map(peakToPeak, 0, 600, 0, maxScale);
```

Now the fun part - time to put the lights on the tree. This is done with a for loop. There are 24 LEDs on the bar graph. If you recall, that is the value of the maxScale constant at the top of the source code. This value is now used as a condition of the for loop - the loop terminates when the maxscale is reached. In the for loop, we go through each LED (represented by the counter, i) and determine what state it should be in based on the displayPeak value we calculated. If our displayPeak value is, say, 12 (representing a sound pressure that makes it “half way” up the bargraph), then the first 12 LEDs on the bar graph will light up, according to this for loop. If the amplitude fills up the bar, then the LEDs turn red, and indicates you should probably be a little more quiet.

```
// draw the new sample
   for (int i = 0; i <= maxScale; i++)
   {
     if (i >= displayPeak)  // turn off these LEDs
     {
        bar.setBar(i, LED_OFF);
     }
     else if(displayPeak > maxScale)
     {
      bar.setBar(i, LED_RED); // bar will flash red when peaking
     }
     else
     {
       bar.setBar(i, LED_GREEN);
     }

   }
   bar.writeDisplay();  // write the changes we just made to the display
}
```

Now, the loop() function runs again from the top. Twenty times in one second! This is fast enough to make the bargraph feel very responsive to sound in the environment. It can easily detect sound from two rooms away.

Click the image below to see it in action (opens a link to YouTube).



Closing

For the video - the audio is a tone played at increasing frequency - I should have played a constant frequency tone and adjusted loudness, for a clearer test, but the effect here is kind of neat. With the matrix graph used in the Adafruit page, I could have tried to generate a waveform based on the sound, rather than just the audio level from the bar graph. I may have opted for a microphone without a built in amp, as it was quite sensitive. Overall the project went quite well, without much trouble.

References

[1] Earl, Bill, and lady ada. “Adafruit Microphone Amplifier Breakout.” *Adafruit Learning System*, Adafruit, learn.adafruit.com/adafruit-microphone-amplifier-breakout/measuring-sound-levels?view=all.